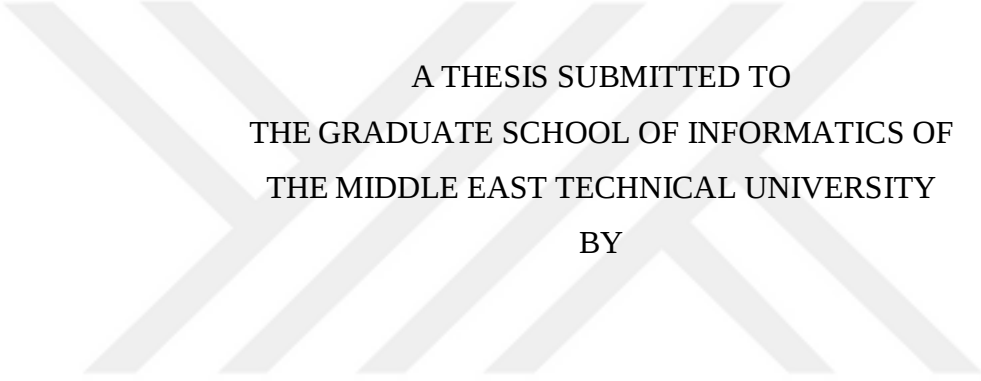


EYE TRACKING FOR INTERACTIVE ACCESSIBILITY: A USABILITY
ANALYSIS OF COMMUNICATION INTERFACES



A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF INFORMATICS OF
THE MIDDLE EAST TECHNICAL UNIVERSITY
BY

KAAN SERGEN ARSLAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN
THE DEPARTMENT OF COGNITIVE SCIENCE

JUNE 2020



I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name: KAAN SERGEN ARSLAN

Signature : _____

ABSTRACT

EYE TRACKING FOR INTERACTIVE ACCESSIBILITY: A USABILITY ANALYSIS OF COMMUNICATION INTERFACES

ARSLAN, KAAAN SERGEN

MSc., Department of Cognitive Sciences

Supervisor: Assoc. Prof. Dr. Cengiz Acartürk

June 2020, 65 pages

At the late stages of ALS, i.e., Amyotrophic Lateral Sclerosis, individuals may lose their ability to speak, eat, move, and even breathe without ventilator support. Alternative modalities of communication, such as gaze, may have a potential to provide a communication interface that may increase the quality of life in cases as such. This study aims at developing interactive interfaces for gaze-based communication, and it reports a small-scale usability analysis for a comparison of the interfaces. In particular, we focus on two dimensions in the experimental investigation: The interface layout design and gaze recording interfaces. The interface layouts include a virtual QWERTY keyboard, an alphabetical one, as well as a cascades menu. The gaze recording interfaces include a wearable eye tracker and a webcam-based eye tracker. More generally, we focus on the role of the simplicity of the interface, the functionality of it, and the usability, which may be three contradicting aspects of interface design and development.

Keywords: Eye-tracking, gaze communication, usability analysis

ÖZ

ETKİLEŞİMLİ ERİŞİLEBİLİRLİK İÇİN GÖZ TAKİBİ: İLETİŞİM ARAYÜZLERİNİN KULLANILABİLİRLİK ANALİZİ

ARSLAN, KAAN SERGEN

Yüksek Lisans, Bilişsel Bilimler Bölümü

Tez Yöneticisi: Doç. Dr. Cengiz Acartürk

Haziran 2020, 65 sayfa

Amyotrofik Lateral Skleroz (ALS) hastalığının ileri aşamalarında bireyler, konuşma, yemek yeme, hareket etme ve hatta ventilatör desteği olmadan nefes alma yeteneklerini kaybedebilmektedir. Göz izleme tabanlı iletişim yöntemleri kullanılarak, bu aşamadaki insanların yaşam kalitesini arttırabilecek bir iletişim arayüzü oluşturmak mümkündür. Bu çalışmada, göz izleme tabanlı iletişim için etkileşimli arayüzlere sahip uygulamalar geliştirilmesi amaçlanmaktadır ve uygulamaların karşılaştırılması için küçük ölçekli bir kullanılabilirlik analizi rapor edilmektedir. Deneysel analizde, arayüz tasarımı ve göz izleme araçları olmak üzere iki konuya odaklanılmaktadır. Arayüz olarak sanal QWERTY klavye, alfabetik klavye ve hiyerarşik bir menü kullanılmaktadır. Göz izleme aracı olarak giyilebilir bir göz izleyici ve web kamerası tabanlı bir göz izleyici kullanılmaktadır. Bu çalışmada daha genel olarak, arayüz tasarımında birbirleri ile çelişen üç özelliğe odaklanılmaktadır: Sadelik, işlevsellik ve kullanılabilirlik.

Anahtar Sözcükler: Göz izleme, göz izleme tabanlı iletişim, kullanılabilirlik testi



DEDICATION

To My Family

ACKNOWLEDGMENTS

I would like to express my sincere appreciation to my supervisor, Assoc. Prof. Dr. Cengiz Acartürk, in this thesis for boundless help, guidance, patience, suggestions, and feedback throughout this project.

I would like to thank Asst. Prof. Dr. Murat Perit Çakır and Asst. Prof. Dr. Erol Özçelik for participating and contributing to the thesis defense jury.

I am indebted to my family for their endless support and encouragement.

I would like to thank ASELSAN Inc. for supporting roles.

I would like to thank my friend Mehmet Caner Sağlam for his mental support.

Last but not least, my special thanks are for my friend Ayşenur Karaaslan for his patience, understanding, and extreme support and love.

TABLE OF CONTENTS

ABSTRACT	iv
ÖZ.....	v
DEDICATION	vi
ACKNOWLEDGMENTS.....	vii
TABLE OF CONTENTS	viii
LIST OF TABLES	x
LIST OF FIGURES.....	xi
LIST OF ABBREVIATIONS	xii
CHAPTERS	
1. INTRODUCTION.....	1
1.1. Motivation	1
1.2. Outline of the Thesis	2
2. BACKGROUND AND LITERATURE REVIEW	5
2.1. Human Computer Interaction (HCI) for Motor Disabilities	5
2.2. Face Detection.....	6
2.2.1. Face Detection Techniques	6
2.2.2. The Haar Algorithm	7
2.2.3. The Haar Algorithm's Implementation on Face Detection Applications	8
2.2.4. OpenCV.....	9
2.2.5. OpenCV Eye Detection Implementations	10
2.3. Eye Tracking Techniques	11
2.4. Eye-tracking for Accessible Communication	12
3. METHODOLOGY PART 1: CASCADE WINDOW APPLICATIONS.....	15
3.1. Initial Prototype.....	15
3.1.1. Pupil Detection Algorithm	15

3.1.2.	The Roles in the Application	17
3.1.3.	Evaluation of the Program by the Author of the Thesis.....	22
3.2.	Cascade Window Application.....	22
4.	METHODOLOGY PART 2: EYE TRACKER APPLICATIONS	25
4.1.	Pupil Labs.....	25
4.1.1.	Pupil Labs Preparations.....	26
4.2.	Pupil Labs Applications	28
4.2.1.	QWERTY Keyboard Eye Tracker Application	28
4.2.2.	Alphabetical Keyboard Eye Tracker Application	34
4.2.3.	Cascade Window Eye Tracker Application	35
4.3.	Summary of the applications.....	36
5.	USABILITY ANALYSIS.....	39
5.1.	SUS Scores for Usability	41
5.2.	Performance Results.....	44
5.3.	Evaluation of the SUS Scores and Experimental Data	47
6.	DISCUSSION	49
7.	CONCLUSION	51
8.	REFERENCES.....	53
	APPENDICES	57
	APPENDIX A	57
	APPENDIX B	59
	APPENDIX C	63
	APPENDIX D	65

LIST OF TABLES

Table 1: System usability scale for applications	42
Table 2: SUS scores given by the participants	43
Table 3: SUS results of the applications	44
Table 4: Results for tests of the Cascade Window Application and the Cascade Window Eye Tracker Application	45
Table 5: Results for tests of QWERTY Keyboard Application and Alphabetical Keyboard Application	46

LIST OF FIGURES

Figure 1: Integral representation of lion image (from Burghard et al., 2004)	9
Figure 2: Pupil under IR light (from Mantiuk et al., 2012, redrawn).	12
Figure 3: Pupil detection algorithm	16
Figure 4: Detected eye window.....	16
Figure 5: Coordinates in eye window	17
Figure 6: Eye window looking left.....	17
Figure 7: Eye window looking right	17
Figure 8: Admin interface of the program	18
Figure 9: Admin picture selection.....	18
Figure 10: Admin interface after selecting six images.....	19
Figure 11: Threshold value calibration	19
Figure 12: 6 calibration steps	20
Figure 13: Calculated coordinates after calibration	21
Figure 14: Program feedback after user selection.....	22
Figure 15: Cascade window calibration part.....	23
Figure 16: Cascade window application interface	23
Figure 17: Approval window of cascade application.....	24
Figure 18: Second window of cascade window application	24
Figure 19: Pupil Core headset (from Pupil Labs, 2019)	25
Figure 20: Correct pupil location on eye camera (from Pupil Labs, 2019)	26
Figure 21: Different Apriltag markers (from Pupil Labs, 2019).....	28
Figure 22: Keyboard with Apriltag tag36h11 marker.....	29
Figure 23: Calibration with Apriltag marker	30
Figure 24: Example output coordinates of the calibration step.....	31
Figure 25: Example output coordinates of the calibration step - normalized	32
Figure 26: Keyboard to coordinate axes mapping	32
Figure 27: Markers for user feedback	33
Figure 28: Alphabetical Keyboard Eye Tracker Application interface	34
Figure 29: Calibration with Apriltag markers	35
Figure 30: Cascade Window Eye Tracker Application.....	35
Figure 31: Setup for tests	40
Figure 32: SUS scores given by the participants (the error bars show 5% of the scores).	43
Figure 33: Average SUS scores of the applications (the error bars show the standard deviation).....	44

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ALS	Amyotrophic Lateral Sclerosis
AOI	Area of Interest
BSD	Berkeley Source Distribution
CR	Corneal Reflection
EEG	Electromyographic
EMG	Electromyographic
FD	Face Detection
GUI	Graphical User Interface
HCI	Human Computer Interaction
IR	Infrared Radiation
OpenCV	Open Source Computer Vision
ROI	Region of Interest
SVM	Support Vector Machine
TCP	Transmission Control Protocol

CHAPTER 1

INTRODUCTION

1.1. Motivation

Selin is 12 years old, and she lives in the village of İstanbul, Turkey. She has physical disabilities that she cannot move her arm, feet, and head. Moreover, she does not have the ability to speak. She goes to the school near her house named Erenköy Elementary School. There are two people in Selin's classroom whose names are Emre and Berk. They do not have the ability to speak, and they cannot move their arms and legs either. Selin, Emre, and Berk have no chance to talk with other kids in the school. Selin's grandfather Mehmet is an ALS (Amyotrophic Lateral Sclerosis) patient. He has the same disabilities as Selin, Emre, and Berk. They face challenges when they need to communicate with others.

Selin, Emre, Berk and Mehmet represent persona that have motor disabilities, as many others diagnosed by ALS, i.e., Amyotrophic Lateral Sclerosis, and relevant disabilities. The statistics show how prevalent ALS is worldwide. The number of ALS diagnosed patients was 222,800 in 2015, which is expected to increase by 69% to 376,674 in 2040 due to the aging of the population (Arthur et al., 2016). The statistics indicate an urgent need for solutions that may contribute to remedy the situation in multiple fronts. Cognitive Sciences have high potential to contribute to the development of research and technologies that accompany humans in their daily life. In particular, the development of AI (Artificial Intelligence) agents as conversational agents will open the way for alternative communication channels for motor-disabled citizens. The goal of this thesis is to contribute to the development of one of those alternative channels, namely gaze-based communication.

Eye-tracking technology is a growing field for interactive and diagnostic applications by detecting eye movements in various domains, such as neuroscience, video gaming, experimental psychology (Mele & Federici, 2012). Recent developments in eye-tracking technologies bring an opportunity to improve the quality of life of ALS diagnosed citizens by providing an alternative communication modality as such. In this thesis we study the potential of gaze-based communication by investigating alternative methods for interface design.

ALS is known as a motor neuron disease, which is caused by gradual degeneration and death of motor neurons. Motor neurons extend from the brain to the spinal cord and muscles throughout the body, meaning that in the later stages of ALS disease, individuals lose the ability to speak, eat, move and even breathe. The potential of computer-based speech synthesizers using eye-tracking technology has already been

recognized as a means to increase the quality of life of citizens with motor disabilities (National Institute of Neurological Disorders and Stroke, 2019). In this thesis, a comparative analysis of alternative interface designs is reported. In particular, we compare the use of an on-screen keyboard vs. a hierarchical window in a simulated setting, where participants with no motor disabilities tested the systems for functionality and usability.

Eye-tracking has already been used in various motor disability contexts. Several decades ago, Hutchinson et al. (1989) described an eye tracking interface for motor-disabled individuals. More recently, Lepu et al. (2012) introduced a software application in which the keywords are selected by the user moving a cursor on a computer screen. The present study aims at contributing to available research by studying the functionality and usability of alternative interfaces. In particular, we report four applications, which are software programs developed for a desktop or a laptop computer. The applications use either a webcam or an eye tracker hardware (in this case, an open-source eye tracker, viz. Pupil Core headset). The applications aim to detect and analyze eye movements and create a communication environment for the individuals.

The underlying research motivation behind the development of the applications is that we recently have limited knowledge about how users allocate attentional resources when they interact with a computer by alternative communication modalities, how they perform actions to accomplish given tasks, and how they perceive the interfaces in terms of their usability.

In this thesis, we investigate the question that is it possible to develop a virtual keyboard environment for communication by gaze that is acceptable from the perspective of usability of the interface? Our hypothesis is that the simplicity of the interface, the functionality of it, and the usability are contradictory aspects of interface design and development. This challenge pertains not only to the interface of the display but also to the hardware. In particular, an affordable solution is a webcam, whereas a costly but effective solution is a devoted eye tracker. In summary, this study aims at evaluating the potential of alternative hardware devices and interfaces for gaze-based communication.

1.2. Outline of the Thesis

Chapter 2 presents the background and literature review for the HCI (Human Computer Interaction) applications for ALS, as well as face/eye detection techniques within the framework of eye tracking methodology. The chapter introduces a rapid object detection algorithm proposed by Viola and Jones (Viola & Jones, 2001). In addition, the usage of this algorithm for pupil detection with OpenCV library is presented. Then eye structure and analysis of eye by an eye tracker is explained.

Chapter 3 describes the usage of Viola and Jones (2001) algorithm. In this thesis, an eye tracker application was developed by using the algorithm and OpenCV library in

the scope of analysis of webcam frames. The program enables a user to communicate with eye movements.

Chapter 4 presents three applications developed by using an open-source eye tracker (viz. Pupil Core headset). The hardware and its usage are introduced. Then, the applications are described. These applications enable individuals to communicate with eye movements with keyboards and pictures.

Chapter 5 reports an experimental investigation of the applications by four non-disabled users. The usability and accessibility analysis of the applications are presented.

Chapter 6 presents a summary of the thesis, a discussion of the findings and the contribution of the thesis to relevant research and application domains. The limitation and future work are also presented in this chapter.



CHAPTER 2

BACKGROUND AND LITERATURE REVIEW

As presented in Chapter 1, four eye-tracking applications were designed and developed in this thesis. One of the applications uses the webcam of the computer, and the other three use an eye tracker hardware. The following sections explain eye-tracking technologies and applications within the context of Human Computer Interaction.

2.1. Human Computer Interaction (HCI) for Motor Disabilities

In the field of HCI, multiple methodologies have been proposed as methods of interaction and communication for users with motor disabilities. The computer control methods use various characteristics of communication channels, such as the ones that employ the functionalities of the users' tongue, eye movements, and brain signals (Salem & Zhai, 1997; Jacob, 1990; Barreto et al., 1999; Barreto et al., 2000). These methods have been used for running multiple system commands for controlling the interface, in particular cursor movement and/or icon selection mechanisms.

A tongue-operated input device, called *Tonguepoint*, is developed to provide an alternative computer input device (Salem & Zhai, 1997). This device is placed into the mouth like a dental night guard or a sports mouthguard. The movement of tongue and bite operation is used for two-dimensional (2-D) cursor movements and selection process, respectively.

The methods that employ physiological sensors, such as Electromyographic (EMG) and electroencephalographic (EEG), transform biosignals into controls for cursor movements (Barreto et al., 1999; Barreto et al., 2000). Not only cursor moves but also `Left-Click` command and an `on/off` switch for the program are implemented via EMG and EEG signals. In the system, there are four bio-electrodes placed above pericranial muscles and above the occipital lobe of the cerebrum. The cursor commands are derived from the natural and voluntary movements of the user's face by using four bio-electrodes.

Besides the methods that employ tongue and biosignals, eye movements have been used for cursor and selection operations in computer interfaces. Gaze tracking applications have been developed either with computer webcams or IR cameras (Wanluk et al., 2016; Liu et al., 2010). In certain cases, where the level of motor functionality is very limited, in particular, when the users have only the ability to move their eyes, gaze tracking applications have the potential to provide an alternative communication channel. Both on-screen keyboards and mouse cursor applications have been used for this purpose.

In terms of the design and development of interfaces as such, a major technical requirement is the detection of the face and the gaze within the face image. The following section introduces face detection technologies, which have also been used by the applications developed within the framework of the thesis.

2.2. Face Detection

Face detection is a computer technology that determines the location and size of a human face in the image (Kumar et al., 2019). There are some challenges in recognizing faces because of various reasons, such as variety in faces, complexities in the background in the digital image, various skin color, ambiguous facial expressions. Due to these drawbacks, eye detection techniques cannot reach the exact and correct results every time. In other words, there may be occurrences of false positives.

Face recognition is a research topic yielding numerous fields and disciplines. Face recognition has a variety of usages in multiple domains, such as bank card identification, access control, mug shots searching, security monitoring, and surveillance (Tolba et al., 2006). The variation in the utilization of face detection comes in a variety of face recognition techniques. Moreover, recently, face detection is not clear-cut, and it is a challenge for computers since it has many variations, such as pose variation, occlusion, illumination. Accordingly, alternative face-detection solutions have been proposed in the literature. The following section describes specific face detection techniques.

2.2.1. Face Detection Techniques

There are two different approaches for face detection techniques, which are feature-based approach and image-based approach (Ranjan et al., 2019 for a review).

In the feature-based approach, the focus is on invariant features such as hair, eye location, nose location. The assumption in the feature-based approach is that humans can detect different faces in an image or an environment so that there must be some features to recognize faces. The significant part of developing and using a feature-based approach is to obtain invariant features which define the face. On the other hand, in image-based approach, the templates are learned from examples of face images. By analyzing face and non-face images, the relevant characteristics are found with the help of statistical analysis. The learned features are applied in the images to detect if there are face and the location of it. Neural-networks, SVM, and Adaboost learnings are examples of image-based face detection approach. Moreover, this technique consumes too much power. In order to reduce computational power, dimensionality reduction is usually carried out (Kumar et al., 2019).

In this thesis, the feature-based approach was selected due to implementation advantages and the requirement of high CPU power consumption in the designed application. The feature-based approach requires feature extraction for object

detection, face detection in our case. For this purpose, Haar-like features are used, as described in the following section.

2.2.2. The Haar Algorithm

Paul Viola and Michael Jones (2001) proposed a machine learning object detection algorithm for visual objects. The algorithm has been implemented on face recognition. Viola and Jones' approach has high detection rates and results faster among the face recognition systems and techniques. The paper widens the horizon of developers constructing a framework for robust and fast object detection. The main and the most crucial property of the Haar cascade approach is to detect faces relatively rapidly. It detected faces at 15 frame per second rate on 384 by 288-pixel images, which was evaluated as high performance in the technology of the 2000s (Viola & Jones, 2001).

The approach has three main contributions to the object detection world. These are:

- Integral image representation has been proposed to compute features. Integral image has been computed from an image using few operations per pixel. This technique results in very fast feature evaluation. It is different from previous techniques such that it does not work with pixel densities of the image.
- A method to construct a classifier has been proposed only by selecting a few essential features using AdaBoost (Schapire & Freund, 1995). A simple modification of the AdaBoost procedure has resulted in fast feature extraction such that each weak classifier has returned with a single feature.
- A method to analyze features in a cascade mode has been represented. The generated features have been combined in a cascade structure. The simple features were examined to detect the region of area of the image, i.e., ROI, and then more complex features are examined in the ROI. This technique has increased the speed of the detector.

Viola and Jones' (2001) approach minimized computation time while achieving a high detection rate. The characteristics of the method give a variety of usage of the algorithm in face applications.

The Haar-like features have been implemented in numerous studies. Wilson (2006) stated that the Haar algorithm had used the change in contrast values between adjacent rectangular groups of pixels rather than the intensity values of a pixel. Wilson also stated that this approach had reduced power consumption.

Another study using the Haar algorithm was proposed to detect a face in images (Mita et al., 2005). Joint Haar-like feature is based on the co-occurrence of multiple Haar-like features. This approach resulted in reducing the error by 37%, and the detector has been 2.6 times as fast as Viola and Jones' (2001) detector while achieving the same performance. In addition to theoretical studies, many applications using the Haar

algorithm have been developed. Some of those applications are presented in the following section.

2.2.3. The Haar Algorithm's Implementation on Face Detection Applications

Human Computer Interaction could be improved by using gesture recognition, which requires facial feature detection and tracking. It has been stated that the development of HCI has increased the number of face and eye-tracking applications (Bradski, 1998). In these applications, the integral representation of the Haar algorithm plays an essential role in detecting face rapidly (Viola & Jones, 2001).

Wilson (2006) stated that integral image, in other words, intermediate representation for the image, has enabled the fast computation of rectangle features. The integral image is an array such that integral image at location (x,y) contains the sum of the pixels above and to the left of point (x,y).

Suppose that $A[x,y]$ is the original image. $AI[x,y]$ is then the integral image. The computation of the integral image is shown in Equation 1.

$$AI[x,y] = \sum_{a \leq x, b \leq y} A(a,b) \quad \text{Equation 1}$$

Wilson (2006) has implemented Haar-like features on computer software. The program of Wilson was trained with two sets of images. The first set contains at least one face to be detected, and on the other hand, other set does not include a face. Three separate classifiers for the images were trained, one for the eyes, one for the nose, and one for the mouth. Wilson's eye classifier implementation has resulted in 93% positive rate and 23% negative rate accuracy.

In addition to computer implementation, the Haar algorithm was implemented on Xilinx Virtex-5 hardware (Cho et al., 2009). The implementation was 35 times faster than software running on a standard computer. Their research concluded that Haar-like feature-based face detection algorithms could produce effective and powerful applications when combined with other technologies.

In addition to human face detection, an algorithm for detection and tracking of animal faces was presented (Burghard et al., 2004). After detecting the animal faces and face components, the program tracks the movements of animals. The application is exemplified in lions. The integral representation of the algorithm is shown in Figure 1.

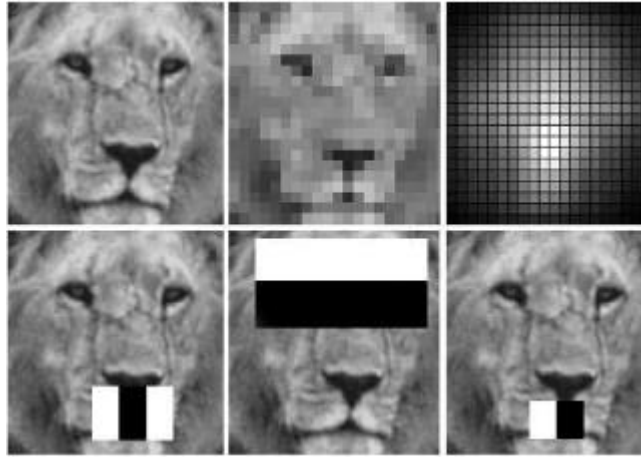


Figure 1: Integral representation of lion image (from Burghard et al., 2004)

The implementation of the Haar-like features is devoted to an open-source library developed by Intel viz. Open Computer Vision Library, i.e., OpenCV.

In this thesis, the Haar algorithm and OpenCV library was used to detect face and eye. A brief explanation of the OpenCV library and the implementations of the library on eye detection are explained in the following sections.

2.2.4. OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source software library with the purpose of computer vision and machine learning. The main purposes of building OpenCV are to provide a platform for computer vision applications and to advance in the usage of machine learning view. OpenCV is a BSD-licensed (Berkeley Source Distribution licensed) product so that OpenCV eases businesses when working with the code. OpenCV library has a wide variety of classic and state-of-art computer vision and machine learning algorithms. More than 2500 optimized algorithms can be used for various purposes, such as recognizing faces, identifying objects, tracking camera movements, tracking moving objects, finding similar images from an image database, removing red eyes from images, and following eye movements. OpenCV library is used extensively in companies, research groups, and by the governments.

Both well-established companies, such as Google, Yahoo, Microsoft, and many startups, such as Applied Minds, VideoSurf, employ the library. OpenCV library has C++, Python, Java, and MATLAB interfaces. The library also supports Windows, Linux, Android, and Mac OS (OpenCV Team, 2019). The features and advantages of the library have resulted in various eye detection implementation, as described in the next section.

2.2.5. OpenCV Eye Detection Implementations

Nguyen and Demidenko (2015) proposed an eye tracker system for drivers to detect drowsiness. The parts of the face are tracked to follow eye blinking frequency, yawn frequency, eye gaze movements, facial expressions, and head movements in the system. These features give information about the drowsiness of the driver. The module tracking the eye and the face movements was implemented in software with the OpenCV library. The module has been driven in PIC 16F887 microcontroller kit. Microcontroller analyzes the frames recorded by the webcam, and the webcam is approximately 20 cm away from the driver's face to capture good images. The output of the algorithm running in PIC 16F887 is sent to the car, and an alarm is generated according to the algorithm output.

Another application for real-time attitude recognition was proposed (Lv et al., 2011). The Haar algorithm on the OpenCV library is implemented to track eye movements. After face and eyes are located, the attitude of the individual has been decided according to the eye shape. This decision is made by the features fed to the algorithm before.

The other application on the OpenCV library, a driver fatigue detection application, was proposed (Li et al., 2011; Nguyen et al., 2015). This method detects the drowsy eyes to set the fatigue driving detection system. In the system, eyes are dynamically tracked to give input to the system.

More recently, an eye-tracking based smart wheelchair design was proposed (Wanluk et al., 2016). The movements of the wheelchair are controlled by the eye movements of the user. In the system, eye tracking was implemented in the imaging processing module called a wheelchair-controlled module. This module contains a webcam to get images of an individual's eye. The frames obtained from the webcam are processed using OpenCV to derive the 2D direction of the eyeball. Then, the movement of the eye is traced to control the movement of the wheelchair.

Similarly, a system for persons with neuro-motor disabilities was developed by arguing that motor disabled citizens could use their eyes for communication. A reliable, mobile, and low-cost system has been presented (Lupu et al., 2013). Webcam and video glasses have been two hardware devices of the Eye-tracking mouse system. In the system, eyeball location is determined using the Haar Cascade Filter in the OpenCV library. After detecting the pupil, the program operates according to the change in location of the pupil. Then the program lets the individual use their eyes as "mouse."

In the applications, the eye-tracking techniques were implemented on visible light. There are other, hardware-based techniques to detect eyes and eye movements (viz. eye trackers). These eye-tracking techniques are explained in the next sections.

2.3. Eye Tracking Techniques

Eye-tracking is a method whereby the position of the eye is used to know where the individual is looking at any given time and sequence in which the eyes of the individual are moved (Poole & Ball, 2006). Since eye tracking is important for researchers trying to understand the movements of the eye while individuals express indifferent actions, different techniques are developed over the years (Lupu & Ungureanu, 2013).

The approaches to determine the locations of the eyeball are categorized into three sets, which are contact lens-based, electrooculogram based, and video-based approaches.

The first approach is contact lens-based eye tracking. A contact lens with mirrors (Yarbus, 1967) or magnetic search coil (Kenyon, 1985) was used in this technique. When contact lens with mirrors was used, a contact lens has been attached to the eyeball. The eye-tracking experiment can last minutes only since the lens can attach to eyeball in a limited time. On the other hand, when magnetic search coil lenses were used, the experiment can last longer. The system in coil lenses operates such that the energy change in eye movements was analyzed to track eye movements. Although there is no time limitation, this approach has not been widely used since individuals need to keep the head stationary in order not to affect power measurements (Lupu & Ungureanu, 2013).

The second approach is electrooculogram-based eye tracking. In this approach, biopotentials of the eye were used to detect eye positions since the movement of the eye causes an electric field. When electrodes are placed near the eye, the change in voltage can be measured to detect eye positions. This technique has not been widely used due to the presence of electrodes on the subject face (Lupu & Ungureanu, 2013).

The third approach is video-based eye-tracking. A video camera was used to track the position of the eye. This approach has two different types that the images are in the visible spectrum, and the images are in the infrared spectrum. Images in visible light processing has some disadvantages such that the process result depends on ambient light. Images in infrared light processing have overcome this problem (Lupu & Ungureanu, 2013). Infrared light eye trackers give more precise results when compared with visible light eye trackers; however, they are more expensive. Infrared light eye trackers apply corneal reflection, i.e., CR, method. When the eyes are exposed to direct IR light, Purkinje image with reflection in the cornea is shown in Figure 2. The corneal reflection is next to the image of the pupil. When the image is observed with an IR spectrum camera, the movement of corneal reflection and pupil can be observed (Mantiuk et al., 2012).

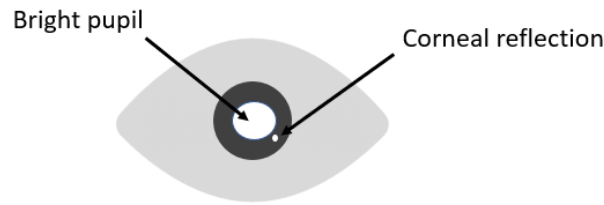


Figure 2: Pupil under IR light (from Mantiuk et al., 2012, redrawn).

In the scope of this thesis, an open-source eye tracker, viz. Pupil Core hardware, was used in order to detect eye location and eye movements. The hardware is an infrared light eye tracker. The usage and developed application of Pupil Core are explained in Chapter 4.

2.4. Eye-tracking for Accessible Communication

There are people with severe motor impairments and neuro-motor disabilities. The people face problems while communicating due to the impairments mentioned in section 1.1. In order to overcome the problems, various eye-tracking applications and equipment are developed.

LC Technologies Eyegaze Inc. is a company specialized in eye-tracking technology. The company has products in the field of eye-tracker devices and software. The company products are used by citizens diagnosed by ALS, brain injuries, spinal cord injuries, and muscular dystrophy. The company has a product viz. *Eyeworld 3.0*. The product is a tablet with an embedded eye tracker. The user of the product can communicate through the on-screen keyboard, browse on the internet, manage e-mails, and use the smartphone by using gaze movements (Eyegaze Edge, 2020). This company also produces eye trackers apart from software development. *EyeDraw* is a software program running on a computer with LC technologies Eyegaze eye tracker. The program enables severe motor disabilities children to draw pictures by gaze movements. The program tracks the gaze movements and transform the gaze data to the drawing. The program checks if the eye position changes or eye is steady for a period. According to the movements of gaze and dwell time, the program makes the corresponding drawings (Hornof & Cavender, 2005). The challenge with this system is its potential cost for the end-user.

Another communication system by gaze movements is the *GazeTalk* system. This system is developed for people who are not able to control their hands (Hansen et al., 2006). The program has e-mail, music player, web browser, and typing interface. The program analyzes the eye movements and dwell time through the selection on the interfaces. The program includes character prediction function on the interface, too. This program is available as freeware in Danish, English, Italian, Chinese and Japanese versions. As of our knowledge, the last update of the *GazeTalk* system was implemented in 2010.

The other eye-tracking application is *EyeAssist*. It is a low cost, user-friendly communication aid developed for citizens with Neuro-Motor Disabilities (Khasnobish et al., 2017). The program uses an eye-tracker hardware of the EyeTribe company (which is obsolete since 2017). The gaze data are acquired from the EyeTribe eye tracker. The EyeAssist program removes the systematic and variable noises of the eye tracker in order to solve the unintentional key presses and extensive calibration process. Kalman filter and linear transformation are used in order to remove the noises in gaze data of the eye tracker. The program has an on-board keyboard with a speech section of the user words. The users of the EyeAssist program found the system to be very user friendly, easy to understand, and learn.

The next assistive application to communicate with gaze movements is developed by using *Tobii EyeX* eye tracker hardware (Pal et al., 2017). The target of the application is to increase the quality of life of citizens with motor disabilities. The application has a "My Need" module for daily needs, SMS API to send messages, an on-screen conversation module, a buzzer sound module, and some mini-modules to improve the life quality of the motor disabled citizens. Like the LC Technologies, this system may also have potentially high cost for the end-user.

In summary, various eye-tracking applications have been developed using different eye trackers. The applications have different user interfaces for users to state their needs. In the present study, we present two different interfaces developed whose are images of daily needs and on-screen keyboard to type needs.



CHAPTER 3

METHODOLOGY PART 1: CASCADE WINDOW APPLICATIONS

This chapter describes the specifications of a webcam-based eye tracking platform that was developed for this thesis. In particular, an eye tracker application with a graphical user interface was designed and implemented. The primary purpose of the program is to communicate with gaze movements.

The program consists of software running on a computer. It processes video frames recorded by the webcam connected to the computer. In the process, face detection, eye detection, and eyeball detection algorithms were implemented. The program detects where the user is gazing at, then operates by tracking the position of the eyes.

In the user interface of the application, six pre-selected pictures are located on the screen. The program tries to detect the area that the user is looking at on the screen. It uses the location and movement of the eyes for the detection process.

Two applications are presented in this chapter: The "Initial Prototype" and the "Cascade Window Application". In the initial prototype, six pictures were located on the screen. However, the eye movements on the vertical axes were not detected efficiently (Section 3.1). Therefore, the prototype was redesigned as the "Cascade Window Application". In this application, only the horizontal positions of the eyes were taken into account, and the vertical positions were ignored. The second application resulted in better performance, as presented in section 3.2.

3.1. Initial Prototype

The program aims to understand where the user is looking at on the computer screen, which is divided into six equal parts. The main part of the program is to detect user pupil location.

3.1.1. Pupil Detection Algorithm

The pupil detection algorithm is shown in Figure 3, which is drawn in drawio (Lupu et al., 2013; Nguyen et al., 2015; Ciesla & Koziol, 2012).

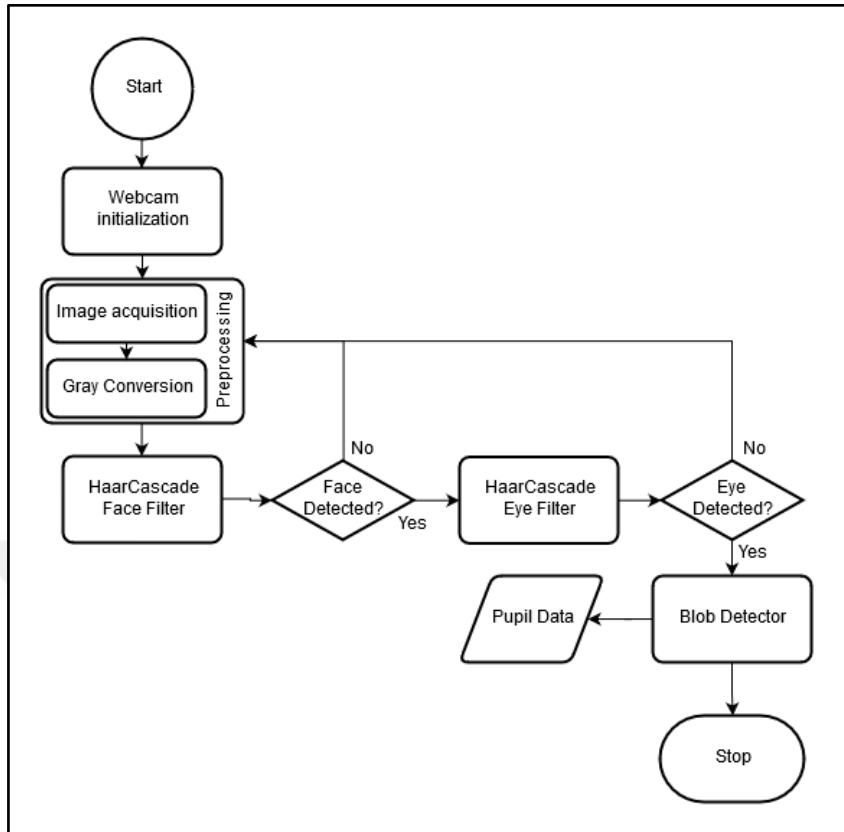


Figure 3: Pupil detection algorithm

At the "HaarCascade Eye Filter" step of the pupil detection algorithm, a window containing an eye is obtained. An example of the eye window is shown in Figure 4.



Figure 4: Detected eye window

The location that the user looks at which part of the screen was decided by the Equation 2 and Equation 3. In the equations, the "rata" value was used to decide the location that the user looks on horizontal axes, and the "ratb" value was used for vertical axes, as shown below.



Figure 5: Coordinates in eye window

$$rata = \frac{x}{a} \quad \text{Equation 2}$$

$$ratb = \frac{y}{b} \quad \text{Equation 3}$$

Figure 6 and Figure 7 show the position of the eyeball on the eye window. As shown in the figures, the "rata" value in Figure 6 is smaller than the "rata" value in Figure 7.



Figure 6: Eye window looking left



Figure 7: Eye window looking right

By analyzing "rata" and "ratb" values, the exact location that the user was looking at was calculated.

The algorithm used in the application to detect pupil location is mentioned. The roles and functions of the application are mentioned in the next sections.

3.1.2. The Roles in the Application

The working principle of the application is that the program shows the user six images and understands which picture the user is looking. The program consists of two sets of roles: An administrator and a user.

- Admin Roles
 - i. Select six images to be located on the GUI
 - ii. Start calibration threshold value for pupil
 - iii. Start calibration computer screen for user
- User Roles
 - i. Select a picture with eyes
 - ii. Approve or disapprove the program's decision

3.1.2.1.Admin Roles

The program starts with admin responsibilities. Admin selects six pictures to be shown to the user. Then, the admin continues to operate with two calibration steps: selecting threshold value and calibrating with respect to the computer screen.

3.1.2.1.1. Selecting Six Images

When the program starts, it welcomes the admin with the interface shown in Figure 8. There are six different buttons for six pictures. Admin selects each of these pictures by clicking the corresponding button, as shown in Figure 9.

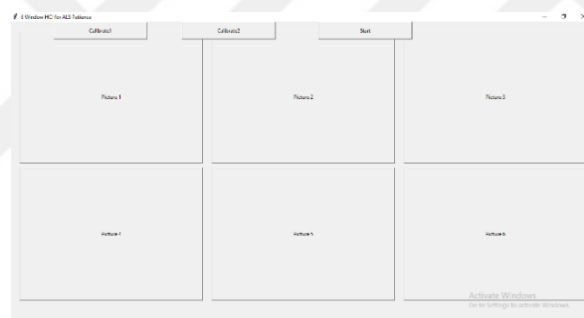


Figure 8: Admin interface of the program

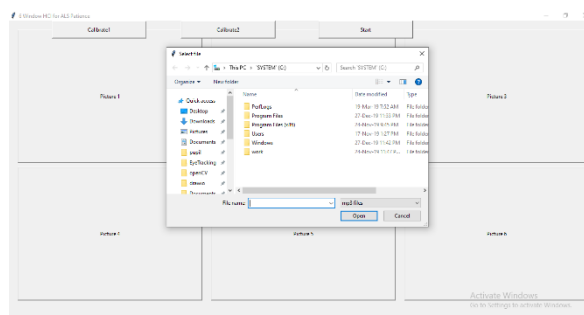


Figure 9: Admin picture selection

After selecting six pictures from the computer, the program is ready to start calibrations. An example of six pictures on the interface is shown in Figure 10.

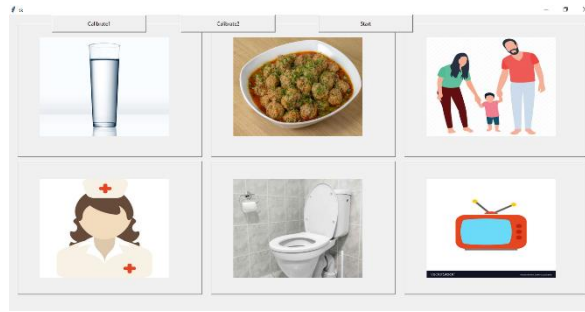


Figure 10: Admin interface after selecting six images

After the pictures are selected, the admin is required to start calibration steps.

3.1.2.1.2. Calibrating Pupil Threshold Value

There are two calibration steps through the program's operation. In the first calibration step, the threshold value for the pupil detection is set.

When pressing the "Calibrate1" button on the interface, video streams recorded by the computer's webcam are observed. Above the video stream, there is a track bar varying between 0 to 255, as shown in Figure 11. The admin user selects the correct value, which is indicated by a red circle on the pupil. When the red circle is stable on the user pupil, then the admin can terminate this calibration step.

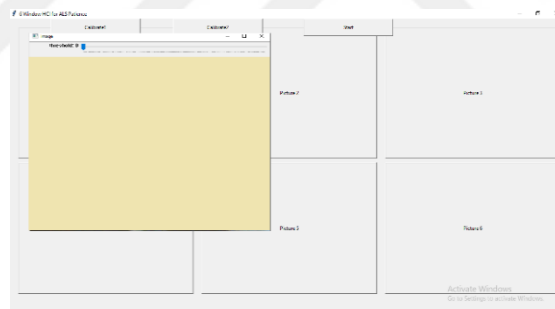


Figure 11: Threshold value calibration

The program continues with the user calibration step.

3.1.2.1.3. Calibrating User Eye for The Computer Screen

The next step is to calibrate the computer screen for the user. In this step, the user is expected to look at the red circle on the computer screen. The program shows the user a screen placing a red circle, as shown in Figure 12. The program collects 50 gaze data in each step.

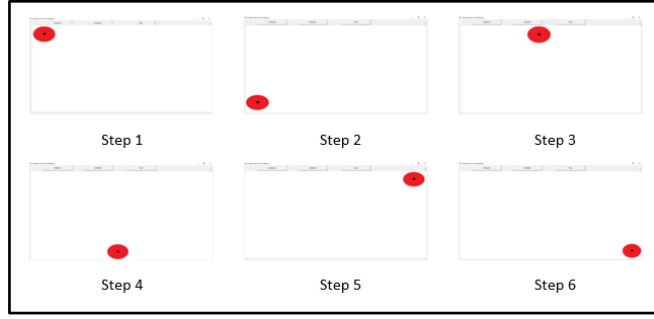


Figure 12: 6 calibration steps

The calibration part consists of six steps, as shown in Figure 12. In every part, 50 "rata" and "ratb" values are collected. In every part of the program, the average values are calculated after ignoring the biggest five and smallest five values. After the calibration part is completed, 12 variables are calculated for later process as follows:

rata₁ and ratb₁ values from Step 1 in calibration.

rata₂ and ratb₂ values from Step 2 in calibration.

rata₃ and ratb₃ values from Step 3 in calibration.

rata₄ and ratb₄ values from Step 4 in calibration.

rata₅ and ratb₅ values from Step 5 in calibration.

rata₆ and ratb₆ values from Step 6 in calibration.

It is time to calculate edges to separate six areas in the program. At the end of the calculation, the edges, three points, a, b and c are obtained, as shown in Figure 13. The calculation of the edges is shown in Equation 4, Equation 5, and Equation 6.

$$a = \frac{rata1 + rata2 + rata3 + rata4}{4} \quad \text{Equation 4}$$

$$b = \frac{rata3 + rata4 + rata5 + rata6}{4} \quad \text{Equation 5}$$

$$c = \frac{ratb1 + ratb2 + ratb3 + ratb4 + ratb5 + ratb6}{6} \quad \text{Equation 6}$$

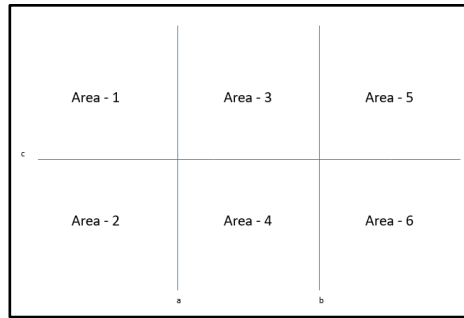


Figure 13: Calculated coordinates after calibration

When this calibration process is completed, the program is ready to operate for the user.

3.1.2.2. User Roles

After the calibration is completed, the program waits for the admin to start. When admin starts the program, a window shown in Figure 10 is presented. The pictures shown in areas would be different according to the admin selection.

The program shows the user six selected picture. The primary role of the user is to select a picture by looking at the picture and then approve or disapprove.

3.1.2.2.1. Selection of Picture

When the admin starts the program, the user is expected to look at one of the pictures. The program collects twenty gaze data from the user and analyzes them to decide where the user is looking at. Twenty gaze data consist of rat_a and rat_b values. The program's calculation is as follows:

- i. Calculate the average of rat_a values and assign the result to rat_1 .
- ii. Calculate the average of rat_b values and assign the result to rat_2 .
- iii. Locate the points of rat_1 and rat_2 values in the coordinate system shown in Figure 13.
- iv. Decide which area contains rat_1 and rat_2 values.

The output of four steps is an area among six ones. After the program decides which area is selected, it shows the user selection with a red rectangle around the image. An example of program feedback is shown in Figure 14.

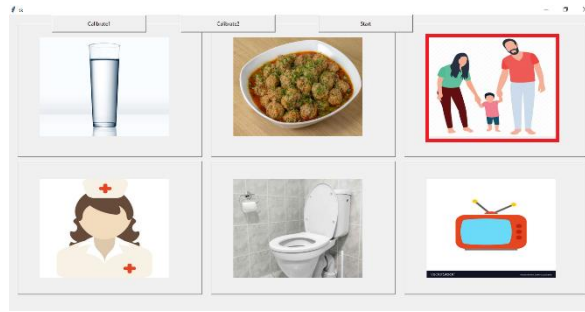


Figure 14: Program feedback after user selection

After this step, the program waits for the user to give a response.

3.1.2.2.2. Approval of the Selection

After the program shows the picture, the program waits for user response. The approval process was handled by closing the eyes. When the program detects the face but not able to detect the eyes, it assumes that the eyes are closed. The program waits for the user for two seconds for approval. When the user closes the eyes for two seconds program concluded that the selection is correct for user desire. Otherwise, the program ignores the user selection.

3.1.3. Evaluation of the Program by the Author of the Thesis

When the gaze data was analyzed, it has been seen that the gaze data on vertical axes did not vary to separate. Moreover, many false positives occurred in test due to the small change of pupil location in vertical axes. The results have led to a new idea in the project, which is the "cascade window" solution.

3.2. Cascade Window Application

The previous program faced with a problem that it fails to detect vertical movements of the eye. To overcome this problem, we developed a solution that the program analyzed only the horizontal movements of the individual's eye. Cascade Window Application was designed for this purpose.

The application detects whether the user is looking at the left side or right side of the screen.

The user role and the screen for the user are different from the initial prototype. In this application, the admin selects 16 pictures.

Before the program is executed, there are two calibration steps. The first calibration step is the same as in Section 3.1.2.1.2. The second calibration step is shown in Figure 15. The user is expected to look left side and right side of the screen, respectively. The calibration process to analyze data is the same with the initial prototype, as explained in Section 3.1.2.1.3.



Figure 15: Cascade window calibration part

When the program initializes, the user encounters the interface, as shown in Figure 16.

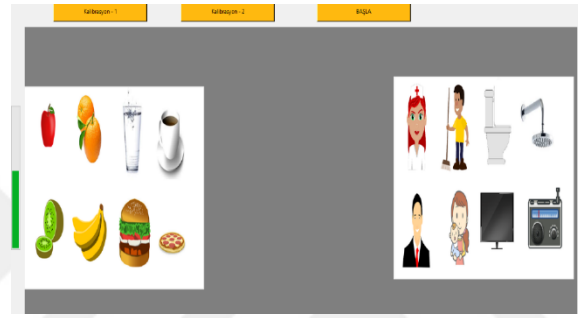


Figure 16: Cascade window application interface

There are sixteen pictures at the beginning of the application. The user is expected to look left or right according to the need. For example, if the user needs "radio", he/she is required to look at the right side of the screen at first.

There are two progress bars on the interface, one is on the leftmost, and the second one is on the rightmost of the screen. At every gaze data, the program decides which side of the screen the user is looking. According to the result, the program updates the progress bar for the user feedback.

After twenty samples are collected, the program analyzes the pupil data of the user. The program decides which side the user is looking at, as explained in Section 3.1.2.2.1. After the selection is finished, the program shows the user only group containing the selected one. Other pictures are removed on the interface, as shown in Figure 17. In the example image, the user looked at the left side of the screen. Then the program waits for the approval or disapproval, as explained in Section 3.1.2.2.2.

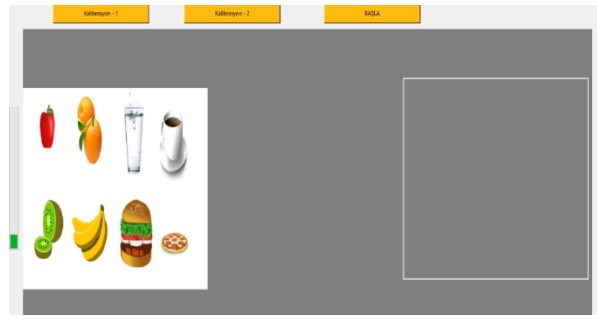


Figure 17: Approval window of cascade application

If the user approves, eight images on the selected side are split four by four, as shown in Figure 18. This process continues until the last picture to be selected.

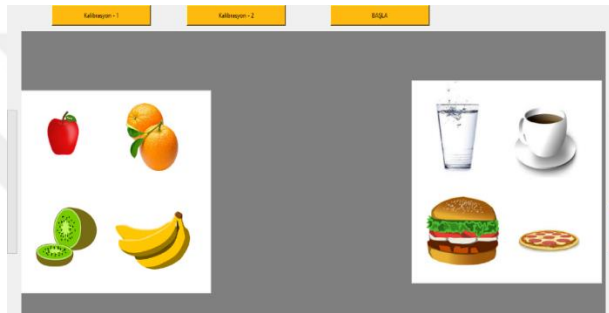


Figure 18: Second window of cascade window application

In this chapter, two applications are mentioned. Due to the problems on the "Initial Prototype", "Cascade Window Application" is designed for better performance. This application is the first application to be used in the tests. Cascade Window Application is the only eye tracker application using visible light in this work. There are three more applications designed to be mentioned in the remaining part of the report. These applications use an eye tracker hardware viz. Pupil Core headset.

CHAPTER 4

METHODOLOGY PART 2: EYE TRACKER APPLICATIONS

The previous chapter described the Cascade Window Application's methodology and usage of it. The application tracks the gaze movements according to the video frames recorded by the computer webcam. Then, the program decides which side of the screen the user is looking and operates according to the result.

However, the major challenge with the Cascade Window application was that the user selection was limited to 16 images. Although the program could be modified for a larger set of images, a keyboard layout provides better flexibility for the user. Therefore, applications with different keyboards were developed in this thesis. Due to the requirements for higher spatial resolution for detecting eye gaze, a feasible equipment that would provide accessibility to the user is an eye tracker. This chapter describes those keyboard applications that were used with an eye tracker. In particular, three applications were developed using open-source eye tracker hardware, viz. Pupil Core Headset.¹

One of the applications has a QWERTY keyboard interface for the user. The other application has a Alphabetical-keyboard interface. The last application's interface is the same as Cascade Window Application. The Cascade Window was modified such that the application works with the eye-tracker hardware. Pupil Core headset preparations and the calibration steps are explained in the following sections.

4.1. Pupil Labs

The Pupil Labs platform is composed of an open-source software suite and a wearable eye-tracking headset. Pupil Core is an open platform that is used by a global community of researchers. In the thesis, the open-source Pupil Core platform was used. The eye-tracking headset is shown in Figure 19 (Pupil Labs, 2019).



Figure 19: Pupil Core headset (from Pupil Labs, 2019)

¹ <https://pupil-labs.com/products/core>

4.1.1. Pupil Labs Preparations

Before operating programs on Pupil Core, necessary programs must be installed, and calibration for the headset must be completed, as explained in the following sections.

4.1.1.1. Setting up the Environment

The required files must be downloaded on the GitHub page of the pupil labs.² Then, the "pupil_capture" executables file must be run.

4.1.1.2. Setting the Headset

i. Wear the Pupil Core

The Pupil Core set consists of a headset and a USB-C connector clip. Pupil Core headset is composed of a world camera, an eye camera, and a nose support. Pupil Core headset is connected to the computer via USB-C to USB-A cable.

ii. Launch Pupil Capture

Once Pupil Capture has initialized, a world video and eye video window appear.

iii. Check Pupil Detection

In the eye video, the pupil needs to be detected. Then, a red circle around the pupil and a red dot at the center of the pupil can be observed in eye video. The good and bad examples of the eye camera outputs are shown in Figure 20 (Pupil Labs, 2019).

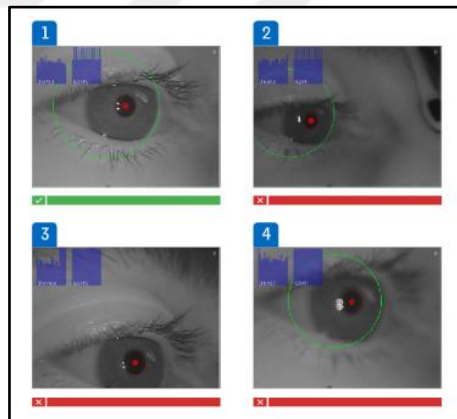


Figure 20: Correct pupil location on eye camera (from Pupil Labs, 2019)

- All range of the eye movements are in the green circle. [Correct]
- The camera arm is not close enough to the eye. [Incorrect]
- The eye is not centered. [Incorrect]

² <https://github.com/pupil-labs/pupil>

- d. The eye is out of focus. [Incorrect]

After the eye location is set, the next step is to observe the pupil values on the world camera. In the left top of the world camera, the confidence needs to be close to 1.00. (In the project, before starting to work, minimum 0.95 confidence was satisfied to obtain good raw pupil data.)

- iv. *Calibrate the Pupil*

When clicking the "c" button on the world window, the calibration starts. In the calibration, the Pupil Capture software shows five different windows, and each one has a marker in different places on the screen. The user is required to look marker for a good calibration.

After setting the environment for the eye-tracker, the plugins must be enabled and configured.

4.1.1.3. Setting the Plugins

In the world video, there is a Plugin Manager menu on the right of the window. In the project, frame publisher, pupil remote, and surface tracker plugins were used.

Pupil Capture has a built-in function to broadcast eye and world camera data on the ZeroMQ library. The following plugins were customized, as explained below:

- i. **Frame Publisher Plugin:**

This plugin broadcasts the video frames from the world and eye cameras. The plugin was enabled and configured to gray so that the plugin broadcasts the grayscale of the video frames.

- ii. **Pupil Remote Plugin:**

This plugin was used to reach Pupil World Camera and Eye Camera data over the local network via TCP protocol. The plugin functions as an entry point to the broadcast infrastructure. The plugin provides control of Pupil Capture over the network.

When the Pupil Remote Plugin is enabled, the program can reach the world and eye video frames over the network.

- iii. **Surface Tracker Plugin:**

This plugin allows users to define a planar surface within the environment to track the Area of Interest. While using this plugin, the surfaces are defined with Apriltag Markers. There are 7 families of Apriltag Markers supported by the Pupil Capture which are tag25h, tag36h11, tagCircle21h7, tagCircle49h12, tagCustom48h12, tagStandart41h12 and tagStandard52h13. The visualization of these markers is shown in Figure 21 (Pupil Labs, 2019).

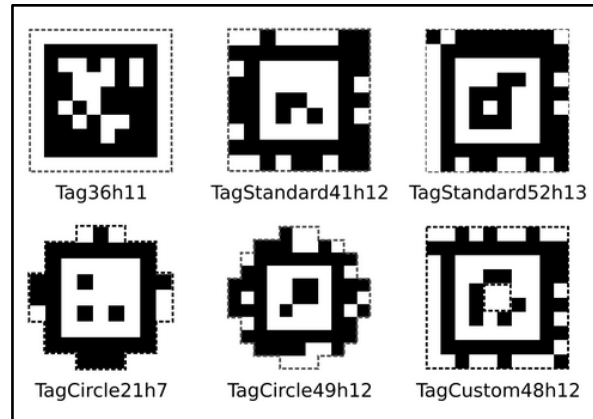


Figure 21: Different Apriltag markers (from Pupil Labs, 2019)

AprilTag is a visual fiducial system. It is useful for a wide variety of tasks, including augmented reality, robotics, and camera calibration (Salem & Zhai, 1997; Jacob, 1990). In this thesis, the AprilTag marker was used to set Area of Interest.

After the preparations are completed, the applications are ready to start. The following sections explain three applications which were developed by open-source Pupil Core headset.

4.2. Pupil Labs Applications

Three applications were designed and developed in this thesis, which used the Pupil Labs eye-tracker hardware. These are:

- i. QWERTY Keyboard Eye Tracker Application
- ii. Alphabetical Keyboard Eye Tracker Application
- iii. Cascade Window Eye Tracker Application

The applications were developed in the python environment.

4.2.1. QWERTY Keyboard Eye Tracker Application

QWERTY Keyboard Eye Tracker Application is the first application designed. The purpose of the application is to communicate via an on-board keyboard. The program detects the point where the eye is looking at on keyboard. Then the program types the corresponding letter and continues with the next letter.

Before starting the program, the plugins must be configured.

After enabling Pupil Remote plugin, the IP address and port numbers were configured. The default IP address is "127.0.0.1," and the default port number is "50020" in the Pupil-Core open-source software. For convenience, the IP and port parameters have remained as default.

The Surface Tracker plugin is operated with a marker, as explained in Section iii. Tag36h11 marker was selected in order to define Area of Interest. Since Surface Tracker Plugin only defines surface with markers, the keyboard picture was combined with the Tag36h11 marker, as shown in Figure 22.



Figure 22: Keyboard with Apriltag tag36h11 marker

The last step is to start calibration in the Pupil Capture program, as explained in Section iv. When the calibration is completed successfully, the Pupil Capture program is ready for the "QWERTY Keyboard Eye Tracker Application" program.

The developed application has four parts to operate. These are:

- i. calibration
- ii. analysis of calibration data
- iii. detecting the letter
- iv. writing the desired phrase.

The parts of the application are explained in the following sections.

4.2.1.1. Calibration

There is a calibration step in the application. It is different than Pupil Capture calibration step. The calibration step of the application maps gaze coordinates to the keyboard. The program shows the user four different pictures, as shown in Figure 23.

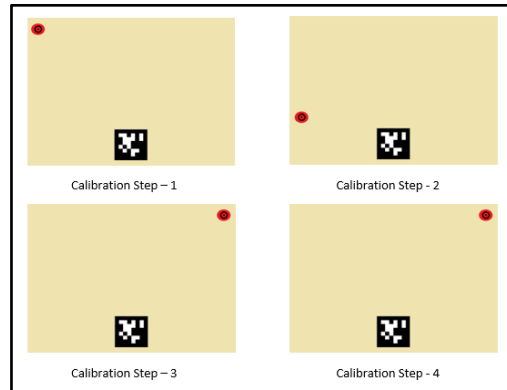


Figure 23: Calibration with Apriltag marker

The program directs the user to look four different corners in the image with a red circle. The program collects 400 pupil data for each corner. When collecting the data, there are two prerequisites: the confidence value must be higher than 0.80, and the gaze direction must be in the region of AIO. After collecting pupil data in the calibration step, the program analyzes the data.

4.2.1.2. Analysis of Calibration Data

There are 1600 gaze data to be analyzed for keyboard mapping.

The following calculations are made for each corner. The output of the calculations are two float values: the first value is x-coordinate, the second value is y-coordinate.

Left-top corner data:

- 1- Append x-coordinates to List_A.
- 2- Append y-coordinates to List_B.
- 3- Sort the values in List_A.
- 4- Remove the first 200 items and last 80 in List A to obtain center values.
- 5- Calculate the average of the remaining 120 items in List_A.
- 6- Assign the value to x1.
- 7- Sort the values in List_B.
- 8- Remove the first 200 items and last 80 in List_B to obtain center values.
- 9- Calculate the average of the remaining 120 items in List_B.
- 10- Assign the value to y1.

Right-top corner data:

Follow the same steps in left corner data by replacing x1 with x2 and replacing y1 with y2.

Left-bottom corner data:

Follow the same steps in left corner data by replacing x_1 with x_3 and replacing y_1 with y_3 .

Right-bottom corner data:

Follow the same steps in left corner data by replacing x_1 with x_4 and replacing y_1 with y_4 .

The program has eight normalized coordinate values, which are x_1 , x_2 , x_3 , x_4 , y_1 , y_2 , y_3 , and y_4 . An example of these normalized coordinates is shown in Figure 24.

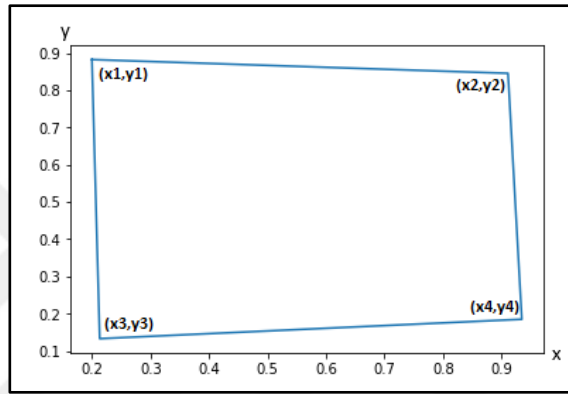


Figure 24: Example output coordinates of the calibration step

The next step is to transform the polygon into a rectangle if the following conditions are satisfied:

- 1- $abs(x_1 - x_3) < 0.1$
- 2- $abs(x_2 - x_4) < 0.1$
- 3- $abs(y_1 - y_3) < 0.1$
- 4- $abs(y_2 - y_4) < 0.1$

These conditions were set in order to ensure that a good calibration was accomplished. If at least one of the conditions is not satisfied, the calibration step is repeated.

The rectangle model of the gaze data is obtained with the calculations, as shown in Equation 7, Equation 8, Equation 9, and Equation 10:

$$xcoord0 = \frac{x_1 + x_3}{2} \quad \text{Equation 7}$$

$$xcoord1 = \frac{x_2 + x_4}{2} \quad \text{Equation 8}$$

$$ycoord0 = \frac{y3 + y4}{2} \quad \text{Equation 9}$$

$$ycoord1 = \frac{y1 + y2}{2} \quad \text{Equation 10}$$

The resulting rectangle generated from sample gaze data is shown in Figure 25.

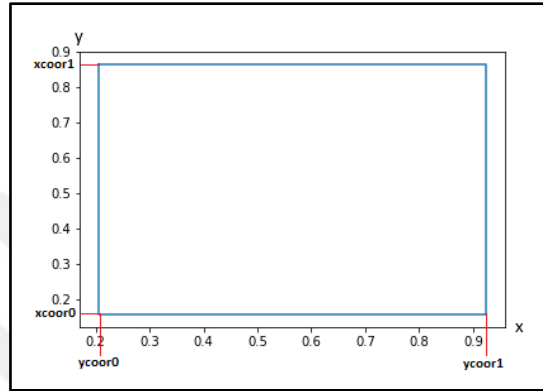


Figure 25: Example output coordinates of the calibration step - normalized

4.2.1.3. Detecting the Letter

So far, the rectangle model was generated. The transform of the keyboard on the rectangle model is shown in Figure 26. The next step is analyzing the gaze data to find the point the user is looking.

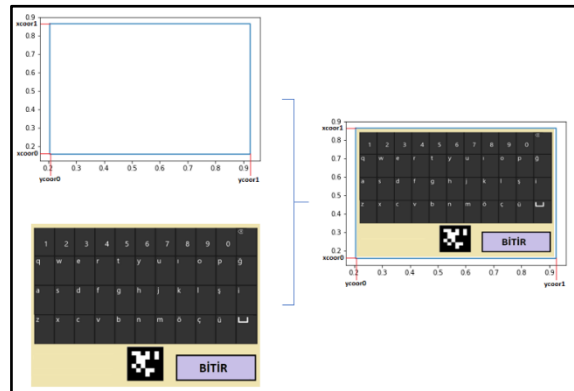


Figure 26: Keyboard to coordinate axes mapping

The program starts with clicking the starting button. When the user triggers the program, the program starts capturing gaze data. The program collects 400 gaze data satisfying that the data confidence value must be greater than 0.80, and in the region of AIO. Then, the application detects the letter which the user is looking.

The same calculations in Section 4.2.1.2 are done to detect the letter. This calculation results in a point (x0, y0) in the coordinate system. Since every letter has numerous points in the coordinate system, the program finds (x0, y0) point belongs in which letter.

The program puts a marker where the user is looking at, as shown in Figure 27 for feedback purposes.

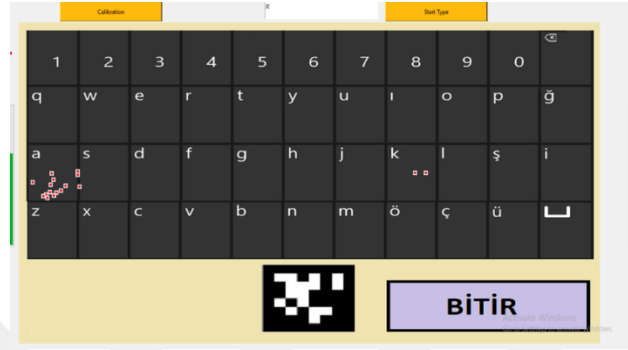


Figure 27: Markers for user feedback

4.2.1.4. Writing the Desired Phrase

When the program decides the letter, it writes to the textbox on the interface, as shown in Figure 27. Then the program continues to obtain the next letter from the user.

The user window also has the following characters: blank and delete.

1. The user can put a space between the words in a sentence or phrase.
2. The user can also delete the last letter if the program types the wrong letter.

After typing by on-screen keyboard, the user is required to finish the program by looking at "BİTİR" (Eng. "end") button.

To sum up, the QWERTY Keyboard Eye Tracking Application is a software program enabling a user to type by looking at the computer screen. In the preparation steps of the program, the Pupil Capture eye and world camera are adjusted for good raw data. Then the necessary plugins of Pupil Capture are activated, and Area of Interest is introduced to the program. After all these prerequisites are satisfied, the program is operated with the calibration step. Then the program collects gaze data to decide where the user is looking at on the keyboard picture. After detecting the letter, the program goes through the next letter until the user finishes the program by looking at the "BİTİR" (Eng. "end") key on the keyboard picture. After the key is pressed by gaze movements, the program is terminated.

Besides the QWERTY keyboard, an application with an alphabetical keyboard was developed. The second eye-tracker application is explained in the following section.

4.2.2. Alphabetical Keyboard Eye Tracker Application

Alphabetical Keyboard Eye Tracker Application is the second application designed with the Pupil Core eye-tracker. The working principle of the application is the same as the application mentioned in Chapter 4.2.1 QWERTY Keyboard Eye Tracker Application. The only difference is the ordering of the letters on the interface. The letters are in alphabetical order. The keyboard interface in this application is shown in Figure 28.



Figure 28: Alphabetical Keyboard Eye Tracker Application interface

In addition to the keyboard applications, a cascade window application was developed. The application is the same interface with the Cascade Window Application, but the Pupil-Core eye-tracker was used to track gaze movements. The following section explains the application's usage and functions of it.

4.2.3. Cascade Window Eye Tracker Application

Cascade Window Eye Tracker Application is the third application designed with the eye-tracker hardware. This application is an alternative one to the application mentioned in Section 3.2 Cascade Window Application. However, in this application, an eye tracker was used to track eye movements instead of computer webcam.

In this application, there is a calibration process the same as the QWERTY Keyboard Eye Tracker Application, as shown in Figure 29.

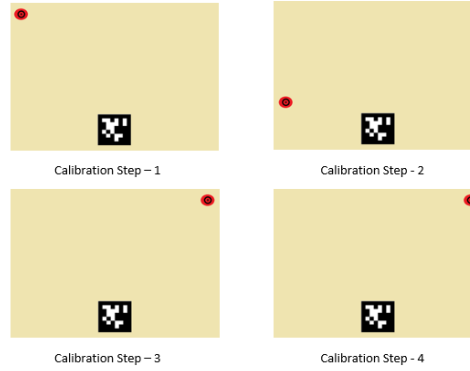


Figure 29: Calibration with Apriltag markers

After the calibration part is completed, the procedures in Section 4.2.1.2 Analysis of Calibration Data was implemented. Then, there is a threshold value from the values $y_{coords0}$, and $y_{coords1}$ are calculated in this application.

$$y_threshold_{threshold} = \frac{y_{coords0} + y_{coords1}}{2} \quad \text{Equation 11}$$

According to the threshold value, the program detects which side of the screen the user is looking at. The program decides if the user is looking left or right by comparing it with the threshold value. In this application, there are "BAŞA DÖN" (Eng. "start from the beginning") and "BİTİR" (Eng. "end") buttons to be clicked, as shown in Figure 30.

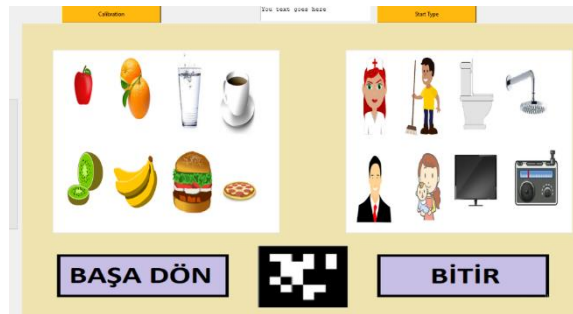


Figure 30: Cascade Window Eye Tracker Application

The program starts with the user clicking the start button. When the user triggers the program, the program starts capturing gaze data. The program collects 400 gaze data satisfying the data confidence value that must be greater than 0.80 and in the region of AIO. The data has two float values, respectively, x coordinates and y coordinates. The collected data needs to be analyzed in order to set coordinates on the screen.

The same calculations in Section 4.2.1.2 are done to detect the letter. This calculation results in a point (x0, y0) in the coordinate system. According to the x0 and y0 values, the program operates as follows:

- If the (x0,y0) coordinates are mapped on the "BİTİR" button, the program stops operating with no output.
- If the (x0,y0) coordinates are mapped on the "BAŞA DÖN" button, the program goes back to the state that the calibration is just completed.
- If the (x0,y0) coordinates are mapped on the left picture, the picture on the left side is divided into two different pictures. These two pictures are located on the left and right sides of the interface. The program goes back to the gaze data collection step.
- If the (x0,y0) coordinates are mapped on the right picture, the picture on the right side is divided into two different pictures. These two pictures are located on the left and right sides of the interface. The program goes back to the gaze data collection step.

If there is only one image on the left side and one image on the right side, the program is in the final state. According to the user gaze data, the program decides the user's needs. Then the program is finished by showing the last picture selected.

4.3. Summary of the applications

Four applications were developed in this thesis. These:

- i. Cascade Window Application
 - A computer camera is used.
 - Gaze data is analyzed under visible light.
 - The user selects one image among 16 ones.
 - The image selection is handled by step by step in a cascade manner.
- ii. QWERTY Keyboard Eye Tracker Application
 - Pupil Core eye tracker is used.
 - Gaze data is analyzed under infrared light.
 - The user writes words/phrases/sentences letter by letter.
 - QWERTY keyboard is used.
- iii. Alphabetical Keyboard Eye Tracker Application
 - Pupil Core eye tracker is used.

- Gaze data is analyzed under infrared light.
 - The user writes words/phrases/sentences letter by letter.
 - Letters are in alphabetical order.
- iv. Cascade Window Eye Tracker Application
- Pupil Core eye tracker is used.
 - Gaze data is analyzed under infrared light.
 - The user selects one image among 16 ones.
 - The image selection is handled by step by step in a cascade manner.

In this chapter, an open-source eye-tracker platform viz. Pupil Core and usage of the eye tracker are mentioned. Three applications are using this eye-tracker platform. Although the user interfaces of the application are different from each other, their main purpose is to build a communication environment by gaze movements.

The next step is to evaluate the usability and performance of these applications. For this purpose, the applications were tested by four participants. The information about the usability analysis is covered in the Usability Analysis Chapter.



CHAPTER 5

USABILITY ANALYSIS

Usability is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use, as defined in ISO 9241 (ISO, 1998). Although there are no specific guidelines on how to measure usability, there are typically two types of usability tests: Formative test and summative test.

Formative tests are performed in order to find and fix as many problems. However, the summative tests are performed in order to describe the usability of an application using metrics. There are two types of summative tests: Benchmark and Comparative. The Benchmark Usability Test aims to evaluate the usability of an application in terms of a set of benchmark goals. On the other hand, Comparative Usability tests are performed in order to compare products or applications.

There are two types of comparative tests: Within-subject design and between-subject design. Whereas the same users work on all products in a within-subject design, different users attempt tasks on each product in a between-subject design. No matter which usability method is selected, data collection is a part of the usability analysis.

Usability data can be collected in traditional sessions such that a moderator observes and reports while user attempts tasks. However, remote moderated and unmoderated sessions have become popular. During these sessions, various data is reported, such as completion rates, task time, number of errors, and satisfaction ratings (Sauro & Lewis, 2016).

In this thesis, a comparative usability test is performed for four applications in order to evaluate and to compare the usability of the applications. All users attempt to complete tasks in each application during tests. During tests, various quantitative data are reported in traditional sessions.

The previous chapters described four eye-tracking applications. One of the applications uses video frames recorded by the computer webcam to obtain gaze data. The other application uses an eye-tracker hardware (viz. Pupil Core). The applications have either a keyboard interface or a hierarchical window interface. In order to evaluate the applications in terms of functionality and usability, an experimental investigation was performed. This chapter describes the usability and performance analysis of these applications.

For the functionality and usability testing, the applications were run either with a computer webcam or with an eye-tracker. There are different preparation steps for different eye-tracking devices, as described below.

Webcam Setup: This environment was set for the test using a computer webcam. As presented in Chapter 3, the Cascade Window Application was run with webcam.

The computer was located at the eye level of the participant, as shown in Figure 31. The computer level is essential in the Cascade Level Application for high-quality data recording.



Figure 31: Setup for tests

Eye Tracking Setup: This environment was set for the tests using the wearable eye tracker hardware. The participant and the computer were placed at the same horizontal location, as in the webcam setup.

The webcam setup and the eye-tracking setup are two preparation steps for the testing. There are two types of user roles: an *admin* and a *user*. The author of the thesis was always in the admin role, and the participants in the user role. The admin initialized the programs and gave tasks to the users in the applications.

There are various tasks for each application. The user was expected to use all four applications twice. The tasks were decided before the test. The tasks were the same for all participants. The tasks are as follows:

Task-1: Ask a glass of water.

Task-2: Call the nurse.

The user was expected to choose the water image in Cascade Window Applications. The user was also expected to write meaningful words or phrases related to the task in keyboard applications.

Four participants (two female, mean age 26, $SD = 2$) with no motor disabilities tested the applications. The applications were used by the participants in different order to

prevent carry-over effects. However, the order of presentation of the tasks was the same. The dependent variables of the study were the success rate, the operation time, the number of errors, and the number of letters.

The collected data was analyzed after data collection from all of the four participants. These data and SUS scores for the applications are valuable to measure the effectiveness and usability of the applications. The following sections explain SUS scores and experimental data analysis.

5.1. SUS Scores for Usability

After tests were completed, every participant answered the standard usability questionnaires for each application. System Usability Scale, i.e., SUS (Brooke, 1996), is the questionnaire conducted. SUS consists of 10 questions, as shown in Table 1. The questionnaires consist of ten items, each with five scale steps. The odd-numbers items have a positive tone; the even-numbered items have a negative tone.

Table 1

System usability scale for applications

The System Usability Scale Standard Version		Strongly disagree			Strongly agree	
		1	2	3	4	5
1	I think I would like to use this application frequently.	☐	☐	☐	☐	☐
2	I found the application unnecessarily complex.	☐	☐	☐	☐	☐
3	I thought the application was easy to use.	☐	☐	☐	☐	☐
4	I think that I would need the support of a technical person to be able to use this system.	☐	☐	☐	☐	☐
5	I found the various functions in this application worked well together.	☐	☐	☐	☐	☐
6	I thought there was too much inconsistency in this application.	☐	☐	☐	☐	☐
7	I would imagine that most people would learn to use this application very quickly.	☐	☐	☐	☐	☐
8	I found the application very difficult to use.	☐	☐	☐	☐	☐
9	I felt very confident using the application.	☐	☐	☐	☐	☐
10	I needed to learn a lot of things before I could get going with this application.	☐	☐	☐	☐	☐

The participants answered all the questions after the test. The SUS scores were calculated as follows (Sauro & Lewis, 2016):

- i. The score contribution of odd numbers of questions (positively worded items), scale position minus 1 (x_i-1)
- ii. The score contribution of odd numbers of questions (positively worded items), 5 minus scale position ($5-x_i$)
- iii. Sum the ten questions' scores.
- iv. Multiply the result by 2.5.

SUS scores for each application are calculated, as shown in Figure 32. For details, see Table 2.

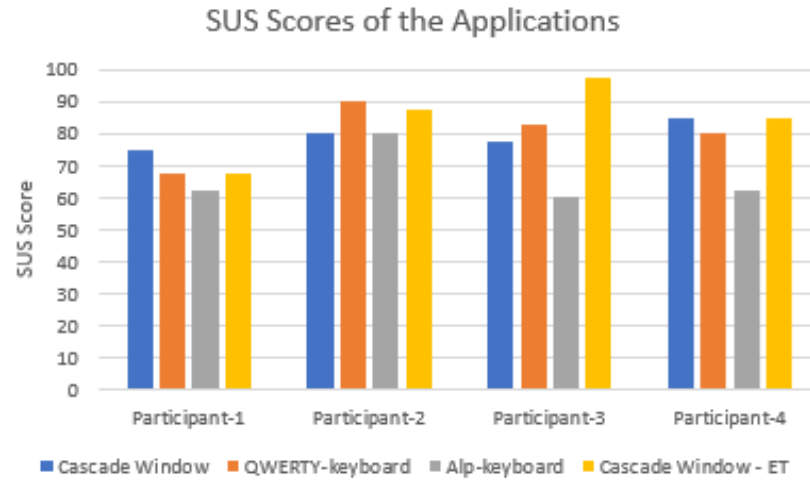


Figure 32: SUS scores given by the participants (the error bars show 5% of the scores).

Table 2 SUS scores given by the participants

SUS scores given by the participants

Participants	Cascade Window	QWERTY Keyboard	Alphabetical Keyboard	Cascade Window - ET
Participant – 1	75.0	67.5	62.5	67.5
Participant – 2	80.0	90.0	80.0	87.5
Participant – 3	77.5	82.5	60.0	97.5
Participant – 4	85.0	80.0	62.5	85.0

In order to evaluate the applications in terms of usability, the SUS scores were analyzed for each application separately, and the averages of the scores were calculated. The results are presented in Figure 33. For details, see Table 3.

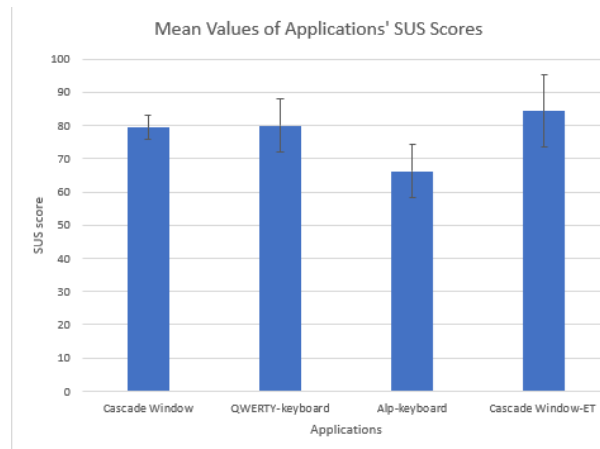


Figure 33: Average SUS scores of the applications (the error bars show the standard deviation).

Table 3 SUS results of the applications

Statistic	Cascade Window	QWERTY keyboard	Alp-keyboard	Cascade Window -ET
N	4	4	4	4
Minimum	75.0	67.5	60.0	67.5
Maximum	85	90	80	97.5
Mean	79.3	80	66.2	84.4
Standard Deviation	7.40	16.2	16.0	21.6

According to the SUS results in Table 3, the usability score of the Alphabetical Keyboard Application is relatively low, though with high variations among the individual scores. The following sections report the performance data of the applications.

5.2. Performance Results

The data set consisted of the following recordings:

- Success rate: It is 0 if the task is not completed and 1 if the task is completed.
- Operation time: The time that the user completes the task.
- Number of errors: The user's bad selections numbers.

- Number of letters: Number of letters used to complete the task. The data is valid in QWERTY Keyboard and Alphabetical Keyboard Applications.

In this section, the results for the Cascade Window Application and the Cascade Window Eye Tracker Application are reported together since they have the same interface. They both have pictures to be selected. The comparison of the results of the cascade window applications gives information about the usability of computer webcam and eye tracker usage. The performance results of cascade window applications are shown in Table 4.

Table 4

Results for tests of the Cascade Window Application and the Cascade Window Eye Tracker Application

Variables	Webcam 1 st task	Webcam 2 nd task	Eye Tracker 1 st task	Eye Tracker 2 nd task
Success Rate	100%	100%	100%	100%
Completion Time - Mean	163.8	109.8	81.8	66.5
Completion Time - Std	70.1	10.9	6.72	15.1
Number of Errors - Mean	1.75	0.25	0	0
Number of Errors - Std	1.92	0.43	0	0

According to the statistics and data in Table 4:

- All the participants have completed their tasks in both applications. As a result, the success rate of the applications is 100%.
- When the completion time is observed, Cascade Window Eye Tracker Application has an advantage over Cascade Window Webcam Application. Participants have reached their goals in a shorter period.
- When the number of errors is observed, Cascade Window Eye Tracker Application has an advantage over Cascade Window Webcam Application.

None of the four participants made a bad selection in Cascade Window Eye Tracker Application.

When completion time and number of errors are taken into consideration, Cascade Window Eye Tracker Application is better than Cascade Window Webcam Application in terms of functionality. The participants felt more comfortable in the selection with Eye Tracker rather than computer webcam since the results of the Eye Tracker is more precise than a webcam.

Following the same line of argument presented above, QWERTY Keyboard Application and Alphabetical Keyboard Application are reported together, since they have keyboard interfaces. The comparison gives information about the usability of the QWERTY keyboard and alphabetical keyboard. Also, the results of keyboard applications are shown in Table 5.

Table 5 Results for tests of QWERTY Keyboard Application and Alphabetical Keyboard Application
Results for tests of QWERTY Keyboard Application and Alphabetical Keyboard Application

Variables	QWERTY Keyboard 1 st task	QWERTY Keyboard 2 nd task	Alphabetical Keyboard 1 st task	Alphabetical Keyboard 2 nd task
Success Rate	100%	100%	100%	75%
Completion Time - Mean	82.8	146.8	151.8	219.3
Completion Time - Std	17.5	30.0	32.4	18.6
Total/Target Letter- Mean	1.5	1.31	2.75	1.39
Total/Target Letter - Std	0.35	0.24	0.75	0.32

According to the results and data in Table 5:

- All four participants have reached their goal in QWERTY Keyboard Application. However, one of the participants was not able to complete one step in the second task of the Alphabetical Keyboard Application.

- The participants have completed their tasks faster in QWERTY Keyboard than Alphabetical Keyboard. The time required to type a word in the QWERTY keyboard is shorter than the alphabetical keyboard.
- For the incorrect letter selection ratio is higher in Alphabetical Keyboard Application. Participants have encountered problems when finding the letters when the letters are ordered alphabetically.

When success rate, completion time, and incorrect letter rate are taken into consideration, QWERTY Keyboard is better than Alphabetical Keyboard in terms of functionality. The participants felt more comfortable in the selection with the QWERTY keyboard rather than the alphabetical keyboard.

In addition to the comparison of the QWERTY Keyboard and Alphabetical Keyboard, the participants get used to finding letters in Alphabetical Keyboard Application. When the first and the second experimental investigation of the application is reported, the error rate in letter selection is decreased in the second test.

5.3. Evaluation of the SUS Scores and Experimental Data

To conclude, Alphabetical Keyboard Eye Tracker Application is the least usable application among four applications according to the usability questionnaire. Moreover, the results show similarities among system usability scores, since the completion time and incorrect letter selection rate is higher when compared to other applications.

Also, Eye Tracker Cascade Window Application is the most useful application in terms of SUS scores, and this application has zero bad selection in the tests. Participants reached their goals without making any wrong choices.

Finally, for the Cascade Window and Eye Tracker QWERTY Keyboard Applications, the SUS scores are around 80, meaning that the applications are usable. Moreover, the experimental results show that participants reached their goal when using these two applications.



CHAPTER 6

DISCUSSION

In the scope of the thesis, four applications were designed, and the usability and functionality of the applications were evaluated as explained in the previous sections. According to the results, we have concluded that it is possible to develop a communication environment by gaze. We have also found that the applications with simple interfaces are more usable.

Beyond these findings, valuable qualitative data is reported during the tests. Firstly, the setup processes for both webcam and eye-tracker applications are challenging for some of the participants. The calibration time for webcam changed between ten seconds to twenty seconds among them. One of the participants' calibration time and selection time lasted longer when compared with others since the application could not detect the eyes of the participant due to frequent blinking.

On the other hand, the preparation time for the eye tracker varies, too. The time spent for the eye tracker preparation step is between forty-five seconds to five minutes. According to the qualitative observation, the size and morphology of the participant eye has an effect on preparation time. Moreover, the calibration time of two of the participants having colored eyes lasted shorter than others in the experiments.

Also, in the keyboard applications, the participants tried to write long sentences for the first task, however, they have seen the difficulty of typing by gaze movements. When they got used to the application, they started to write shorter phrases to express their needs. Moreover, the participants figured out how the applications obtain best gaze data. They accomplished that by observing the programs feedback such that progress bars on cascade window applications and red dots on keyboard applications.

Another observation of the author of the thesis was that some of the participants had selected the incorrect letters in the keyboard applications. These participants gave feedback such that the letters could be grouped in order to increase the resolution, which could prevent participants from making the incorrect selection.

As mentioned in the Introduction chapter, the target group of the applications are motor disabled people, however, the usage of the applications could be upgraded to the people not having motor disabilities. There are various examples of the products that developed for disabled people but used by non-disabled people such as a powered toothbrush. In the tests, some of the participants enjoyed in typing on-screen keyboard. They had requested to use the application after tests are completed. This may be an instance of a situation in which the technology of disabled citizens are used by non-disabled citizens.

Moreover, the World Health Organization announced coronavirus disease as a pandemic in the time that this thesis was written. According to the announcements, it

is possible that a person can get infected by touching a surface or object that has the virus on it and then touching their mouth, nose, or possibly their eyes, this is not thought to be the main way the virus spreads (Cucinotta & Vanelli, 2020). According to the announcement, touching a surface is a way of the virus to spread, and managing the actions by gaze movements rather than touching would be trendy in the future works.

Beyond the findings, analysis, and implementation of the applications, further research is needed to establish a more accurate and user-friendly communication environment. The gaze data analysis method including a fixation detection algorithm would be developed for better performance. Moreover, the auto-complete module would be included in the user interface to write the words faster in the keyboard applications.

Finally, we believe that the design and development of this thesis have indicators that contribute to improving the quality of life of physiology patients.

CHAPTER 7

CONCLUSION

As mentioned in the introduction chapter of the report, there are numerous ALS diagnosed people worldwide, and their population is expected to increase in years. Since ALS diagnosed people face challenges when they need to communicate with others, there is an urgent need for design, development and testing novel communication methods for usability, as well as functionality on alternative user interfaces.

Since eye-tracking technology is a growing field for interactive and diagnostic applications, gaze-based communication solutions bring an opportunity to improve the quality of life of ALS diagnosed citizens. Gaze-based communication solutions has already been used in various motor disability context such as ALS. This study aims at contributing to available research by studying the functionality and usability of alternative interfaces.

In this thesis, we reported four gaze-tracking applications, which are developed for a desktop or a laptop computer. The aim of the applications is to detect and analyze eye movements and create a communication environment for the individuals.

Throughout the thesis, we seek an answer for the research question that is it possible to develop a virtual keyboard environment for communication by a gaze that is acceptable from the perspective of usability of the interface. The evaluation of the potential of alternative hardware devices and interfaces for gaze-based communication is covered in the report. The eye-tracking techniques, implementation of the techniques, designed applications, and evaluation of the applications in terms of usability and functionality are covered in this report.

In chapter 2, various eye-tracking techniques are introduced. The implementation and usage of these techniques in the HCI application are mentioned. Viola and Jones's rapid object detection algorithm and usage of the algorithm for pupil detection are introduced in the chapter (Viola & Jones, 2001). Moreover, the structure of the eye image and the analysis of the gaze location by an eye tracker are explained in this chapter.

Chapter 3 represents the implementation of the Viola and Jones algorithm by analyzing frames recorded by a webcam. An eye tracker application using the algorithm is introduced in this chapter. The program enables a user to communicate with eye movements. In the application, it is observed that eye movements on vertical axes are hard to discriminate. As a result, only the movement of the eye on horizontal axes is analyzed.

In chapter 4, an open-source eye-tracker Pupil Core environment and the applications using the hardware are introduced. The applications enable individuals to

communicate with eye movements with onboard keyboards and hierarchical window interfaces. The applications are differentiated in terms of the interface simplicity, mainly.

In chapter 5, the usability and functionality of the applications are evaluated. The data collected during the experimental investigation and usability analysis are investigated in this chapter, too. According to the experimental investigation, the users found the system to be user friendly and easy to understand. The feedback mechanism in the tests enables users to understand and learn the applications while using them.

To sum up, we proposed four eye-tracking applications using different hardware to detect pupil movements on different interface that user interacts. We provided a usability analysis for the applications. According to the experimental data and usability scores, the Alphabetical Keyboard Application is relatively low, though with high variations among the individual scores.

Moreover, the eye-tracker cascade window application is better than webcam cascade window application in terms of functionality, since eye-tracker hardware is more precise than a webcam. Similarly, the QWERTY keyboard interface is more usable than the alphabetical keyboard interface.

The usability and functionality of the applications are evaluated by four participants. The number of participants could be increased for a better evaluation. Beyond the evaluation of the applications, the author of the thesis observed that the setup process for the eye tracker is not a simple process for the users. Some of the participants face difficulties in the preparation step of the eye-tracker. To reduce the difficulties of the eye tracker usage on individuals can be subjects of future studies. Moreover, calibration marker implementation would be used to increase the face detection rate and speed in the webcam applications for future works.

In addition to these difficulties, the author of the thesis observed that the calibration was lost when the participants move their head too much. Area of Interest could be changed to reduce calibration lost in future studies. Moreover, in order to reduce false-positive rates in the keyboard applications, a fixation detection algorithm can be implemented for future works. For fast and accurate detection of gaze data, marker calibration technique would be used in webcam applications. Beyond a fixation detection algorithm and marker calibration, the setup for the usage of applications have to be simplified for ALS diagnosed people.

REFERENCES

- Amyotrophic Lateral Sclerosis (ALS) Fact Sheet. (2019, August 13). Retrieved October 17, 2019, from <https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Fact-Sheets/Amyotrophic-Lateral-Sclerosis-ALS-Fact-Sheet>.
- Arthur, K. C., Calvo, A., Price, T. R., Geiger, J. T., Chio, A., & Traynor, B. J. (2016). Projected increase in amyotrophic lateral sclerosis from 2015 to 2040. *Nature communications*, 7(1), 1-6.
- Barreto, A. B., Scargle, S. D., & Adjouadi, M. (1999). A real-time assistive computer interface for users with motor disabilities. *ACM SIGCAPH Computers and the Physically Handicapped*, (64), 6-16.
- Barreto, A., Scargle, S., & Adjouadi, M. (2000). A practical EMG-based human-computer interface for users with motor disabilities.
- Bradski, G. R. (1998). Computer vision face tracking for use in a perceptual user interface.
- Brooke, J. (1996). SUS-A quick and dirty usability scale. *Usability evaluation in industry*, 189(194), 4-7.
- Burghardt, T., Calic, J., & Thomas, B. T. (2004, October). Tracking Animals in Wildlife Videos Using Face Detection. In *EWIMT*.
- Chauhan, M., & Sakle, M. (2014). Study & analysis of different face detection techniques. *International Journal of Computer Science and Information Technologies*, 5(2), 1615-1618.
- Ciesla, M., & Koziol, P. (2012). Eye pupil location using web-cam. *arXiv preprint arXiv:1202.6517*.
- Cho, J., Mirzaei, S., Oberg, J., & Kastner, R. (2009, February). Fpga-based face detection system using haar classifiers. In *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays* (pp. 103-112).
- Cucinotta, D., & Vanelli, M. (2020). WHO declares COVID-19 a pandemic. *Acta bio-medica: Atenei Parmensis*, 91(1), 157-160.
- Eyegaze Edge. (2020). Retrieved May 16, 2020, from <https://eyegaze.com/products/eyegaze-edge/>

- Freund, Y., & Schapire, R. E. (1995, March). A decision-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory* (pp. 23-37). Springer, Berlin, Heidelberg.
- Hansen, J. P., Lund, H., Aoki, H., & Itoh, K. (2006). Gaze communication systems for people with ALS. In *ALS Workshop, in conjunction with the 17th International Symposium on ALS/MND, Yokohama, Japan*.
- Hornof, A. J., & Cavender, A. (2005, April). EyeDraw: enabling children with severe motor impairments to draw with their eyes. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 161-170).
- Hutchinson, T. E., White, K. P., Martin, W. N., Reichert, K. C., & Frey, L. A. (1989). Human-computer interaction using eye-gaze input. *IEEE Transactions on systems, man, and cybernetics*, 19(6), 1527-1534.
- International Organization for Standardization. (1998). *ISO 9241-11: Ergonomic requirements for office work with visual display terminals (VDTs): Part 11: Guidance on usability*.
- Jacob, R. J. (1990, March). What you look at is what you get: eye movement-based interaction techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 11-18).
- Jones, P., Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *University of Rochester. Charles Rich*.
- Kenyon, R. V. (1985). A soft contact lens search coil for measuring eye movements. *Vision research*, 25(11), 1629-1633.
- Khasnobish, A., Gavas, R., Chatterjee, D., Raj, V., & Naitam, S. (2017, March). EyeAssist: A communication aid through gaze tracking for patients with neuro-motor disabilities. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* (pp. 382-387). IEEE.
- Kumar, A., Kaur, A., & Kumar, M. (2019). Face detection techniques: a review. *Artificial Intelligence Review*, 52(2), 927-948.
- Li, X., Han, G., Ma, G., & Chen, Y. (2011, November). A new method for detecting fatigue driving with camera based on OpenCV. In *2011 International Conference on Wireless Communications and Signal Processing (WCSP)* (pp. 1-5). IEEE.
- Liu, S. S., Rawicz, A., Ma, T., Zhang, C., Lin, K., Rezaei, S., & Wu, E. (2010). An eye-gaze tracking and human computer interface system for people with ALS and other locked-in diseases. *CMBES Proceedings*, 33.

- Lupu, R. G., & Ungureanu, F. (2013). A survey of eye tracking methods and applications. *Buletinul Institutului Politehnic din Iasi, Automatic Control and Computer Science Section*, 3, 72-86.
- Lupu, R. G., Ungureanu, F., & Bozomitu, R. G. (2012, August). Mobile embedded system for human computer communication in assistive technology. In *2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing* (pp. 209-212). IEEE.
- Lupu, R. G., Ungureanu, F., & Siriteanu, V. (2013, November). Eye tracking mouse for human computer interaction. In *2013 E-Health and Bioengineering Conference (EHB)* (pp. 1-4). IEEE.
- Lv, Y., Wang, S., & Shen, P. (2011, August). A real-time attitude recognition by eye-tracking. In *Proceedings of the Third International Conference on Internet Multimedia Computing and Service* (pp. 170-173).
- Mantiuk, R., Kowalik, M., Nowosielski, A., & Bazyluk, B. (2012, January). Do-it-yourself eye tracker: Low-cost pupil-based eye tracker for computer graphics applications. In *International Conference on Multimedia Modeling* (pp. 115-125). Springer, Berlin, Heidelberg.
- Mele, M. L., & Federici, S. (2012). Gaze and eye-tracking solutions for psychological research. *Cognitive processing*, 13(1), 261-265.
- Mita, T., Kaneko, T., & Hori, O. (2005, October). Joint haar-like features for face detection. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1* (Vol. 2, pp. 1619-1626). IEEE.
- Nguyen, T. P., Chew, M. T., & Demidenko, S. (2015, February). Eye tracking system to detect driver drowsiness. In *2015 6th International Conference on Automation, Robotics and Applications (ICARA)* (pp. 472-477). IEEE.
- Olson, E. (2011, May). AprilTag: A robust and flexible visual fiducial system. In *2011 IEEE International Conference on Robotics and Automation* (pp. 3400-3407). IEEE.
- OpenCV. (2019, December 23). Retrieved from <https://opencv.org/>.
- Pal, S., Mangal, N. K., & Khosla, A. (2017, March). Development of assistive application for patients with communication disability. In *2017 international conference on innovations in green energy and healthcare technologies (IGEHT)* (pp. 1-4). IEEE.
- Poole, A., & Ball, L. J. (2006). Eye tracking in HCI and usability research. In *Encyclopedia of human computer interaction* (pp. 211-219). IGI Global.

- Pupil Labs. (2019). Retrieved November 20, 2019, from <https://pupil-labs.com/>.
- Ranjan, R., Bansal, A., Zheng, J., Xu, H., Gleason, J., Lu, B., ... & Chellappa, R. (2019). A fast and accurate system for face detection, identification, and verification. *IEEE Transactions on Biometrics, Behavior, and Identity Science*, 1(2), 82-96.
- Salem, C., & Zhai, S. (1997, March). An isometric tongue pointing device. In *Proceedings of the ACM SIGCHI Conference on Human factors in computing systems* (pp. 538-539).
- Sauro, J., & Lewis, J. R. (2016). *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann.
- Tolba, A. S., El-Baz, A. H., & El-Harby, A. A. (2006). Face recognition: A literature review. *International Journal of Signal Processing*, 2(2), 88-103.
- Viola, P., & Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, 57(2), 137-154.
- Wang, J., & Olson, E. (2016, October). AprilTag 2: Efficient and robust fiducial detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 4193-4198). IEEE.
- Wanluk, N., Visitsattapongse, S., Juhong, A., & Pintavirooj, C. (2016, December). Smart wheelchair based on eye tracking. In *2016 9th Biomedical Engineering International Conference (BMEiCON)* (pp. 1-4). IEEE.
- Wilson, P. I., & Fernandez, J. (2006). Facial feature detection using Haar classifiers. *Journal of Computing Sciences in Colleges*, 21(4), 127-133.
- Yarbus, A. L. (1967). Eye movements during perception of complex objects. In *Eye movements and vision*. (pp. 171-211) Springer, Boston, MA.

APPENDICES

APPENDIX A

Voluntary Participation Form

This study was conducted by METU Cognitive Science graduate students Kaan Seren Arslan. It is carried out within the scope of a master's thesis under the supervision of Assoc. Dr. Cengiz Acartürk. This form has been prepared to inform you about the research conditions.

What is the purpose of the study?

The purpose of this study is to obtain information about the usability of the use of Eye Tracking Communication Applications.

How are we going to ask you to help us?

The research is going to be performed in the METU library. Working adults are going to be invited as participants and the test is going to take 20 minutes. In the study, you are going to be asked to perform the tasks by eye movements.

What you need to know about your participation:

Participating in this study is voluntary. You can refuse to participate or stop working without being subject to any sanction or penalty. You can leave blank if you have questions that you do not want to answer during the research.

The data collected from the participants of the research is going to be kept completely confidential and the data and identity information are not going to be matched in any case. The names of the participants are going to be collected in a separate list. In addition, only researchers can access the collected data. The results of this research can be used in scientific and professional publications or for educational purposes, but the identities of the participants are going to be kept confidential.

Risks:

During work, you may feel physical discomfort while looking at the computer screen. You can stop the procedure at any time and stop working without any loss. In such a situation, it is going to be sufficient to tell the person who applied the study that you want to quit the study. The activity of looking at the computer screen is a frequently used method in research, and it does not pose any significant risk.

If you want to get more information about the research:

You can send your questions and comments about the study to the researcher at kaan.s.arslan@gmail.com.

I have read the above information and fully voluntarily participate in this study.

(After completing and signing the form, return it to the practitioner).

Name Surname

Date

Sign

---/---/----

APPENDIX B

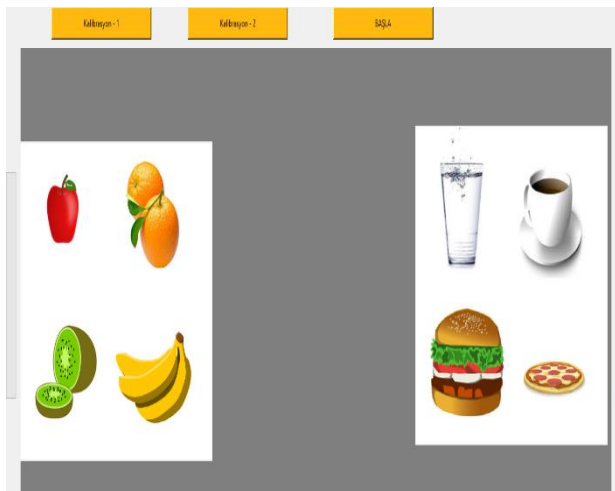
Pre-test Information Form

I would like to thank you for participating in my work voluntarily. The purpose of this test is to test the usability of the designed programs. Programs are not intended to measure your skills. You can leave the test at any time

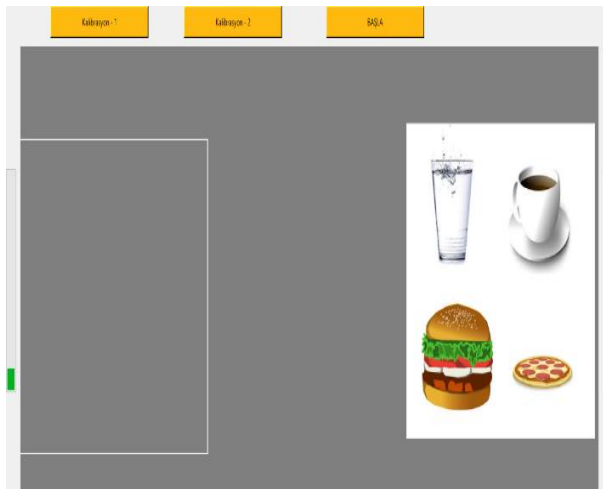
Please read the following warnings and recommendations carefully to guide you during the tests:

- 1- Since these programs are designed for ALS patients who have lost speech and mobility, it is very important that you do not speak and move after the test begins.
- 2- In this study, you are going to use 4 eye-tracking applications.
- 3- One of the programs is done with the camera of the computer, the remaining three are made with the "pupil core" hardware. No equipment used in the study has a detrimental effect on human health.
- 4- In two of the programs you are going to use, I am going to ask you to choose one of the 16 photos shown on the screen.
- 5- In two of the programs you are going to use, I am going to ask you to write the specified expression with the keyboard you are going to see on the screen.
- 6- There are calibration procedures during the running of the programs. I am going to support you in these transactions.
- 7- After the programs are completed, you need to fill in the questionnaire where you can send your feedback to me.

Application – 1:



- After the calibration is completed, you are going to see this interface.
- In which part of the screen (right / left) you want to select, look there.
- The progress bars next to the photos indicate that the program understands that you are looking there.
- When the progress bars are full, you are going to be switched to the other screen.



- When switching to this screen from the above screen, a notification sound comes from the computer.
- After this sound is heard, if the selection of the program is the same as your selection, close your left eye and confirm the selection. If the program selection is wrong, just keep your eyes open.
- Open your eyes and continue after a second notification sound comes from the computer.

Application – 2:



- After the calibration is completed, you are going to see this interface.
- You must write the word in this program letter by letter.
- The program marks where the user is looking.
- After a letter is written, the program shows the letter to the user in the upper part of the interface.
- There is a delete key on the top right when you think you have typed the wrong letter.
- After you have finished writing, look at the text "BİTİR" (Eng, "end").

Application – 3:

This program works in the same way as Program-2. In the program, the letters are arranged alphabetically instead of the QWERTY keyboard.

Uygulama – 4:



- After the calibration is completed, you are going to see this interface.
- In which part of the screen (right / left) you want to select, look there.
- The program marks where the user is looking.
- When the wrong choice is made, the selection is withdrawn by looking at the word "BAŞA DÖN" (Eng. "start from the beginning").
- When you want to exit the program, see the text "BİTİR" (Eng. "end").

Calibration information note:

There is a calibration step before you start using the programs. I am going to start the calibration step by informing you. The round mark "  " shown on the screen should be followed during the calibration step.



APPENDIX C

System Usability Scale (SUS)

This is a standard questionnaire that measures the overall usability of a system. Please select the answer that best expresses how you feel about each statement after using the applications today.

	Strongly Disagree	Somewhat Disagree	Neutral	Somewhat Agree	Strongly Agree
1. I think I would like to use this application frequently.	☐	☐	☐	☐	☐
2. I found the application unnecessarily complex.	☐	☐	☐	☐	☐
3. I thought the application was easy to use.	☐	☐	☐	☐	☐
4. I think that I would need the support of a technical person to be able to use this system.	☐	☐	☐	☐	☐
5. I found the various functions in this application worked well together.	☐	☐	☐	☐	☐
6. I thought there was too much inconsistency in this application.	☐	☐	☐	☐	☐
7. I would imagine that most people would learn to use this application very quickly.	☐	☐	☐	☐	☐
8. I found the application very difficult to use.	☐	☐	☐	☐	☐
9. I felt very confident using the application.	☐	☐	☐	☐	☐
10. I needed to learn a lot of things before I could get going with this application.	☐	☐	☐	☐	☐

How likely are you to recommend this website to others? (please circle your answer)

Not at all likely 0 1 2 3 4 5 6 7 8 9
10 Extremely likely



APPENDIX D

Qualitative Questions

- 1- When the application starts, how did you complete the task?

- 2- Did you feel comfortable when approving the selection?

- 3- Did you feel comfortable when disapproving of the selection? (if applicable)

- 4- Overall, what's your experience been with the application?

- 5- If you could change one thing about the application, what would it be? Why?

- 6- How likely are you to refer to this application? Why or why not?

- 7- What did you think of the layout of the content?

- 8- Which of these two approaches/options do you find best? Why?